2012

Extension of graph clustering algorithms based on SCAN method in order to target weighted graphs

Anton Chertov
University of Windsor

Extension of graph clustering algorithms based on SCAN method in order to target weighted graphs

by

Anton Chertov

A Thesis
Submitted to the Faculty of Graduate Studies
through School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2012

Extension of graph clustering algorithms based on SCAN method in order to target weighted graphs

by

Anton Chertov

APPROVED BY:

_____

Dr. Myron Hlynka, External Examiner
Mathematics and Statistics Department

_____

Dr. Luis Rueda, Internal Reader
School of Computer Science

_____

Dr. Scott Goodwin, Co-Advisor
School of Computer Science

_____

Dr. Ziad Kobti, Co-Advisor
School of Computer Science

_____

Dr. Boubakeur Boufama, Chair of Defence
School of Computer Science

27 January 2012

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

I. Co-Authorship Declaration

I hereby declare that this thesis incorporates material that is result of joint research, as follows: this thesis incorporates the outcome of a joint research undertaken in collaboration and under supervision of Dr. Ziad Kobti and Dr. Scott Goodwin. The collaboration is covered in Chapter 2 of the thesis. In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were performed by the author, and the contribution of co-authors was primarily through the provision of guidance and supervision.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Declaration of Previous Publication

This thesis includes 1 original paper that have been previously published/submitted for publication in peer reviewed journals, as follows:

Thesis Chapter	Publication title/full citation	Publication status*
Chapter 2	Weighted scan for modeling cooperative group role dynamics. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 17 –22, aug. 2010	*published*

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

In this thesis we evaluate current neighbour-based graph clustering algorithms: SCAN, DHSCAN, and AHSCAN. These algorithms possess the ability to identify special nodes in graphs such as hubs and outliers. We propose and extension for each of these in order to support weighted edges. We further implemented two graph generating frameworks to create test cases. In addition we used a graph derived from the ENRON email log. We also implemented a Fast Modularity clustering algorithm, which is considered as one of the top graph clustering algorithms nowadays. One of three sets of experiments showed that results produced by extended algorithms were better than one of the reference algorithms, in other words more nodes were classified correctly. Other experiments revealed some limitations of the newly proposed methods where we noted that they do not perform as well on other types of graphs. Hence, the proposed algorithms perform best on social graphs with pronounced community structure.

DEDICATION

I dedicate this work to my family: my parents Nina and Mikhail and my brother Andrew. This work would never be done without their love and support.

ACKNOWLEDGEMENTS

It is an honour to thank people who made this work possible.

Great thanks to my co-advisor Dr. Scott Goodwin, who made my acceptance to University of Windsor possible and provided me with guidance in University of Windsor and Canada in general.

My sincere thanks to my co-advisor Dr. Ziad Kobti for his endless energy and guidance through my research work.

Deepest thanks to Graduate Secretary Mandy Turkalj for her patience in answering my countless organizational questions.

Also I would like to thank the most important people who contributed to my research: my family. Thanks to my brother Andrew I happened to be accepted to University of Windsor. Thanks to my parents Nina and Mikhail I was able to see this world and take my place in it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A graph, or network, is a structure formed by a set of vertices (or nodes) and set of edges connecting the vertices. This mathematical abstraction can be used to model various real-world structures of interest to modern scientific applications. Graph clustering, or partitioning, involves identifying groups of vertices in a graph that are tightly connected to each other within a group, or has a high level of similarity of some kind, and are weakly connected to vertices from other groups or are dissimilar to them. Further we will use the words 'graph' and 'network', 'clustering and 'partitioning', 'node' and 'vertex', 'cluster' and 'community' interchangeably. Graph clustering has a great importance in almost any field of modern science including biology, geology, geography, computer science, engineering and social sciences. The latter is of specific interest to us. In social networks, clustering enables the modeling and identification of hidden structures (communities) that might not be seen to the observer or even to their members.

I.1. Existing graph clustering algorithms

Great amount of graph clustering algorithms exist today and one can find a lot of information about them in survey on graph clustering by Schaeffer [10]. The deeper insight of the graph clustering and related problems is given by Santo and Fortunato in [28]. Graph clustering is a very vast topic, but in many cases the central idea behind it is defining the similarity function between vertices of the graph. The further grouping of vertices into communities is based on the value of this function, which defines whether vertices are similar enough to be grouped together. For the purpose of social network

analysis, the notion of vertex neighbourhood seems very important to us. Intuitively people sharing many friends are very likely to know each other and thus to be connected. For that reason we would like to pay attention to algorithms that use adjacent vertices as information for similarity functions: SCAN (Structural Clustering Algorithm for Networks) [35] and two more algorithms derived from it: DHSCAN (Divisive Hierarchical Structural Algorithm for Networks) [36] and AHSCAN (Agglomerative Hierarchical Structural Algorithm for Networks) [37]. Designed by the same group of scholars, these algorithms are based on structural similarity function which utilizes information about vertex neighbourhood structure. Some of the mentioned algorithms are also peculiar for their ability to detect vertices that play special roles in graphs: hubs (connect different clusters, but don't belong to any certain one) and outliers (isolated vertices). In social network analysis (especially in epidemiology, marketing, etc.) this knowledge may be crucial.

I.2. Current research motivation

The initial form of SCAN and derived algorithms target only unweighted graphs, which we identify as a serious limitation to their application. For instance, distance between individuals (or degree of their attraction to each other) may be represented by a weight of the edge connecting them. The further the individuals are located from each other, the larger is the weight, the weaker is their ability to communicate (or as the attraction increases, the interaction becomes easier). In [4] a weighted version of SCAN which allows it to overcome the mentioned limitations was introduced. However, the new algorithm was just presented, and was never compared to any other existing clustering

algorithm. Moreover, the formula for structural similarity used for Weighted SCAN at the same time introduces a limitation: similarity between vertices that are not connected by an edge turns to zero, which nullify usage of the mentioned social structure concept.

I.3. Thesis contribution

In this work we proposed modifications to Weighted SCAN (which also can be considered as a separate extension of SCAN) in order to improve its ability to detect clusters and propagate this approach to extend DHSCAN and AHSCAN. Our modification extended the notion of structural similarity so that it became capable of utilizing edge weights. We proposed new clustering algorithms and clustering quality measurement function based on new structural similarity value. The proposed approach was expected to leave algorithms complexity with no changes while making SCAN-based algorithms applicable to weighted graphs. In chapter III we show that algorithmic complexity indeed did not change.

In this work we used planar l-partition model for generating graphs with known a priori community structure, thus making it possible to validate our algorithms. We also designed our own framework for random graph generation. The framework is capable of producing various types of random graphs that possess certain properties of real world graphs:

- preferential attachment [1]
- power laws [3], [11]
- community structure (in edge weights) [3]

To prove the validity of the approach we applied new weighted algorithms to a number of various data sets, including synthetic and real-world graphs. We also implemented Fast Modularity clustering algorithm, introduced by Clauset et. al in 2004 [5], which is considered one of the fastest graph clustering algorithm by time of Xu's publication in 2007 [35], and used it as a reference algorithm. We expected our algorithms to be able to identify community structure in weighted graphs and be competing to the reference algorithm.

Ideally a data set would have community (cluster) structure which is known a priory, so that clustering results produced by the algorithms can be compared to a real community distribution - planted l-partition model serves that purpose. But when community structure is not known a priori, clustering quality functions should be used. We implemented three various clustering quality functions, which focus on various aspects of clustering, to evaluate the results of clustering:

1. Newman's modularity [27], extended to deal with edge weights.

2. Novel Similarity-Based Modularity Function [14] designed by Feng in collaboration with the authors of SCAN.

3. Performance [2].

We also implemented a random 'clusterer', that produces totally random partitions and was supposed to provide the 'ground' level for the graph clustering quality functions values. Partitions, produced by various clustering algorithms were compared in terms of quality functions values. Thus we expected our algorithms to produce graph partitions, which quality, measured by the mentioned functions, would be higher or comparable to

the one of the partitions produced by reference Fast Modularity algorithm. Where possible we also applied visual analysis.

I.4. Thesis outline

The main purpose of this research is to find out if the proposed approach of the structural similarity extension is valid and yields meaningful results. Structural similarity extension consequently influences clustering algorithms that use it and similarity based modularity - clustering quality measurement - and we try to note all the consequences in this study. The research paper is divided into following chapters.

In chapter II we discuss related work in the field of graph clustering, quality functions and random graph generation. We also mention the importance of weights and describe the reference weighted graph clustering algorithm and discuss SCAN algorithm and its derivatives in details.

In chapter III we introduce our approach which makes it possible to utilize edge weights in computations of the weighted version of SCAN. We discuss the influence of weights on the quality functions and random graph generators. We explain our experimental setup and the expected results.

Chapter IV presents the experiment results and provides their analysis. We discuss technical aspects of our experimental setup. The chapter contains three subsections devoted to different types of experiments held.

Chapter V concludes the research, explains insights received during the work, provides the drawbacks of the algorithms and sets up the field of opportunities for the future work.

CHAPTER II

REVIEW OF LITERATURE

II.1. Graph clustering

Network clustering is also known as graph partitioning, which is a task of splitting a graph into a number of sub-graphs that are also called clusters. Given a graph G = {V, E}, where V is a set of vertices (or nodes), E is a set of edges that connect vertices, clustering is partitioning of G into k disjoint sets of vertices $G_i = \{V_i, E_i\}$, such that:

1. $V_i \cap V_j = \emptyset$ for any i and j: $i \neq j$;

2. $V = \bigcup_{i=1}^{k} V_i$

The number of clusters k may be or be not known a priory.

Network clustering, as well as more general clustering problems, has been studied in many science and engineering disciplines for a long time. Recent and commonly used algorithms are of particular interest here. The deeper insight on various approaches and algorithms can be gained from a survey on graph clustering by Elisa and Schaeffer published in 2007 [10].

There is no widely accepted definition of a clustering task. Different algorithms use different definitions and views, but all of them use similar notion that vertices in one cluster are somehow similar to each other being at the same time distinct from the vertices in other clusters.

One of the most well known graph clustering algorithms is the min-max cut method proposed by Ding et al. in [8], which partitions a graph into two clusters. A disconnecting set of a graph G is a collection of edges such that every chain connecting vertex a with vertex b in G meets the collection. A cut is a disconnecting set, 'no proper

subset of which is disconnecting' [16]. The main idea behind this method is to minimize the number of connections between clusters and maximize the number of links within every cluster. The set of edges that needs to be removed to isolate two clusters is called a cut. With regard to this cut-definition, clustering that aims to find optimal cut is an NP-hard problem [29]. The method searches for the minimized cut and tries to maximize the number of remaining edges. This method has a number of drawbacks, that are listed by Yuruk et al in [36], for instance cutting one vertex from the graph achieves an optimum, so in practice some constraints must to be applied to the clustering, but such constraints are not always suitable. A list of practices was introduced to solve this problem, including normalized cut [31]. Still, these approaches provide partitioning only to produce two clusters. To create a partition into *k* clusters one should continue splitting into two clusters achieved from the previous step until *k* clusters are produced. This method, however, does not guarantee the optimality of the result. Moreover *k* is usually unknown.

Newman and Girvan in [27] proposed modularity as a measure of the quality of a network clustering and is defined as:

$$Q_n = \sum_{s=1}^{k} \left[\frac{l_s}{L} - \left(\frac{d_s}{2L} \right)^2 \right] \quad (1)$$

where *k* is the number of clusters in the graph, *L* is the number of edges in the graph, l_s is the number of edges between nodes within cluster *s*, and d_s is the sum of the degrees of the nodes in cluster *s*. When modularity is maximized the optimal clustering is achieved. Modularity was claimed to turn to zero either when there is only one cluster that contains the whole graph or in the case when nodes are clustered at random. Exhaustive search through possible partitions aimed to find the maximum modularity will 'take at least an

exponential amount of time' and thus is not feasible for most of real-world applications [26].

In [26] Newman proposed a greedy optimization algorithm as an optimization method for solving the problem of finding the maximum modularity. It is based on a hierarchical agglomeration and thus falls into the category of agglomerative hierarchical clustering methods [12]. In the beginning every vertex of the graph is placed into a singleton cluster. At every step algorithm merges two clusters. The merging candidates are chosen in order to maximize the modularity increase (or minimize the decrease). The algorithm results are represented in a form of a dendrogram, cuts through which produce graph partitions. The partition with the largest value of modularity is expected to be the best. The algorithm running time is $O((n+m)n)$, where n is number of nodes and m is number of edges in the graph.

Producing satisfactory partitions, this Newman's algorithm remained rather time-consuming and Clauset et al. in [5] proposed an optimization to it based on maintaining and updating a matrix of modularity changes rather than recalculating modularity and maintaining the adjacency matrix. The Clauset's optimization also make use of advanced data structures that speed up the algorithm running time. For today if is one of the fastest clustering methods (as claimed by Xu et al. in [35]) and one of the most cited in the research community. Its running time is $O(md\log(n))$ where n is the number of vertices in a graph, m is the number of edges and d is the depth of the hierarchical cluster structure dendrogram. Considering properties of many real-world networks: sparsity and hierarchy that imply $m \sim n$ and $d \sim \log n$, the algorithm complexity turns into $O(n \log^2 n)$.

This method by-turn has its individual drawbacks. For instance, it fails to recognize hubs and outliers and uses for clustering only structural information derived from the graph.

Another well known and widely used technique for graph clustering is spectral clustering. It is a set of methods that are based on eigenvectors calculation for the matrices that represent the graph structure (most often Laplacians are used for this purpose). Despite the popularity of spectral methods they are considered computationally demanding [10] and fail to recognize nodes in a graph that have special roles (like hubs and outliers) which we see as a very important information. Hubs connect (bridge) clusters and do not belong to any of them. For instance they are key figures in spreading ideas and diseases in social and biological networks. Outliers have only a weak connection with some cluster and may represent hermits in social networks.

II.1.1. SCAN

For the purposes of optimal network clustering together with identifying and isolating hubs and outliers Structural Clustering Algorithm for Networks was developed by Xu et al. and proposed in [35]. SCAN algorithm utilizes the structural similarity as a similarity function, which uses information on the neighbourhood of two connected vertices. In other words, the algorithm is based on common neighbours, that is to say two vertices are assigned to a cluster according to the number of neighbours they share. The algorithm targets simple undirected and unweighted graphs. The authors of SCAN build a solid mathematical base to define the task of clustering based on the formal definitions they introduce. Formal definitions are described below for reference [35]:

Definition 1 (Vertex structure)

Let $v \in V$, the structure of υ is defined by its neighbourhood, denoted by Γ(υ)

$$\Gamma(v) = \{\omega \in V \vee (v,\omega) \in E\} \cup \{\omega\} \tag{2}$$

The absolute amount of neighbours is not very descriptive, so the normalized value of common neighbours is introduced:

Definition 2 (Structural similarity).

$$\sigma(v,\omega) = \frac{|\Gamma(v) \cap \Gamma(\omega)|}{\sqrt{|\Gamma(v)||\Gamma(\omega)|}} \tag{3}$$

Any non isolated vertex in the graph is identical to itself in terms of structural similarity: $\delta(v, v) = 1$.

Definition 3 (ε-neighbourhood).

$$N_\varepsilon = \{\omega \in \Gamma(v) \mid \sigma(v,\omega) \geq \varepsilon\} \tag{4}$$

Structural similarity of two vertices will be large if they share a similar structure of neighbours. A minimum (threshold) value of structural similarity ε is introduced by this definition.

Definition 4 (Core)

Let $\varepsilon \in \Re$ and $\mu \in \mathrm{N}$. A vertex $v \in V$ is called a core with reference to ε and μ, if its ε-neighbourhood contains at least μ vertices:

$$CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon| \geq \mu \tag{5}$$

When a value of structural similarity exceeds the ε-threshold value for enough vertices in a neighbourhood, the central vertex of this neighbourhood becomes a seed or a nucleus for the cluster. SCAN formal terminology identifies such a node as a core. Core is a special type of vertex that has at least μ neighbours with a structural similarity exceeding the threshold ε. Clusters are grown around the cores. When a vertex is in the ε-

neighbourhood of a core it should belong to the same cluster. The value μ is another one (along with ε) of the algorithms parameters and corresponds to a minimum cluster size. In the most cases it equals to 2. Though due to some conditions or case study logic (for instance when agents are supposed to be united in groups of n) it may have different value.

Definition 5 (Direct structure reachability)

$$DirREACH_{\varepsilon,\mu}(\upsilon,\omega) \Leftrightarrow CORE_{\varepsilon,\mu}(\upsilon) \wedge \omega \in N_{\varepsilon}(\upsilon) \quad (6)$$

Direct structure reachability is symmetric for any pair of cores; but if one of the vertices is not a core it is asymmetric.

A formal mathematical theory for clustering is built by Xu et al. in [35]. It is based on aforementioned definitions and some other provided in their work. The authors also formulate and prove a couple of lemmas not mentioned here for brevity. Based on these preliminaries the SCAN algorithm is introduced. The pseudocode of the algorithm is given below and taken from [35]. At the beginning all vertices are marked as unclassified. Each vertex is classified as either a member of a cluster or non-member. Every unclassified vertex is checked whether it is a core (Step 1). If it is a core, a new cluster is expanded from it (Step 2.1), otherwise the vertex is marked as a non-member (Step 2.2). A new cluster is built from the core υ (any if possible) by finding all structure-reachable from υ vertices. New cluster ID is generated to be assigned to all the vertices found in step 2.1. In step 3 non-member variables are classified as hubs or outliers.

```
Algorithm SCAN (G=<V, E>, ε,μ)
// All vertices are unclassified;
```

for each unclassified vertex $\upsilon \in V$ do

// Step 1. Check whether υ is a core;

if $CORE_{\varepsilon,\mu}(\upsilon)$ then

// Step 2.1 if υ is a core, a new cluster is expanded;

generate new clusterID;

insert all $x \in N_{\varepsilon}(\upsilon)$ into queue Q;

while $Q \neq 0$ do

y = first vertex in Q;

$R = \{x \in V \mid DirREACH_{\varepsilon,\mu}(y,x)\}$;

for each $x \in R$ do

if x is unclassified or non-member then

assign current clusterID to x;

if x is unclassified then

insert x into queue Q;

remove y from Q;

else

// Step 2.2 if υ is not a core, it is labelled as non-member

label υ as non-member;

end for.

// Step 3. further classifies non-members

for each non-member vertex υ do

if $\exists x, y \in \Gamma(\upsilon)(x.clusterID \neq y.clusterID)$ then

```
            label υ as a hub
  else
            label υ as outlier;
end for.
end SCAN
```

Figure 1 - SCAN pseudocode

During its execution, the SCAN algorithm queries the neighbourhood of every vertex. Thus the overall complexity is directly proportional to the sum of all the vertices' degrees. It follows that every edge must be counted twice: once from each end. So the algorithm running time is *O(m)*, where m is the number of edges. In the case when the number of edges is unknown the authors provide a complexity analysis for the number of vertices. In the worst case, when we have a complete graph, the complexity is $O(n^2)$, where *n* is the number of vertices. Fortunately, full graphs do not exist often in real life. Hence in the average case of random graphs that have been successfully applied for the models of real networks SCAN complexity is *O(n)* [35]. Xu et al. refer to studies of many biological and non-biological networks applying SCAN and revealing a complexity not exceeding *O(n)*.

The major limitation and drawback of the SCAN algorithm is the fact that for a successful execution, it requires two parameters: ε and μ that may be hard to determine by the user.

II.1.2. DHSCAN

Analogously to the modularity by Newman and Girvan in [27] Feng et al. proposed a novel similarity-based modularity function in [14] defined as:

$$Q_s = \sum_{i=1}^{NC} \left(\frac{IS_i}{TS} - \left(\frac{DS_i}{TS} \right)^2 \right) \quad (7)$$

where *NC* is number of clusters, IS_i - total similarity of vertices in *i*-th cluster, DS_i- total similarity between any vertices in the graph and vertices in cluster *i*, *TS* - total similarity between any two vertices in the graph; $\sigma(u, v)$- structural similarity of vertices *u* and *v*, defined before. We will describe this function in more detail in the next section, devoted to quality measurements. Feng et al. claim that their similarity-based modularity produces better results (compared to classical modularity, which failed to identify hubs and outliers) in graphs with 'more confused' sub-graph structure, including vertices of special importance: hubs and outliers [14]. In the same paper the authors introduce a genetic algorithm claimed to be able to find clusters as well as hubs in the graphs. As a drawback of the genetic algorithm the authors mention its inability to scale to large graphs.

Later the same year after introducing SCAN, Yuruk et al. from the same research group proposed Divisive Hierarchical Structural Clustering Algorithm for Networks (DHSCAN) in [36]. DHSCAN is based on the same principle of structural similarity and it does not require any input parameters from the user, targeting same kinds of graphs: undirected and unweighted. DHSCAN is capable of finding hierarchical structure of clusters in the graph. This algorithm uses the same theoretical base as SCAN, particularly the definitions of the vertex structure and structural similarity. It also extends the theoretical base with a new definition of the edge structure, provided below for reference [36]:

Definition (Edge structure)

Let $\upsilon,\omega \in V$ and $e = (\upsilon,\omega) \in E$, the structure of the edge e is defined by the structural similarity of υ and ω and denoted by:

$$\kappa(e) = \sigma(\upsilon, \omega) \quad (8)$$

The working idea of this algorithm is the difference in edge structure of the inter-cluster and intra-cluster edges. Edges connecting vertices from different clusters have a larger edge structure than inter-cluster edges. Considering that the algorithm sorts the edges in an ascending order of edge structure it removes them one by one, then measures the quality of achieved clusters, and chooses partition with the highest quality. The algorithm uses Feng's similarity-based modularity mentioned earlier in this section to find the optimal partition among many possible options.

The pseudocode of the algorithm is given below and taken from [36]:

```
Algorithm DHSCAN (G = <V, E>)
// in the beginning all edges are classified as intra-cluster ones
W := E; B := Φ; i := 0; Q_i := 0;
while W≠ Φ do {
// Move edge with minimal structure;
remove e := min_struct(W) from W;
insert e into B;
find all connected components in W;
if (number of components increased) {
        i := i + 1;
```

```
        define each component in W as a cluster;
        plot level i of dendrogram;
        calculate Q_i
}
}
// Get the optimal clustering;
cut the dendrogram at maximal Q value;
end DHSCAN.
```

Figure 2 - DHSCAN pseudocode

The authors point the ability to identify the optimal partition and the absence of any input parameters as the advantages of their algorithm. None of the limitations or drawbacks are mentioned as well as no complexity analysis is provided. Meanwhile the long computation time is the main disadvantage of the algorithm as the demanding operation of modularity calculation has to be applied every time the clustering structure changes after the removal of certain number of edges.

II.1.3. AHSCAN

Some time after proposition of SCAN and DHSCAN an agglomerative version of the hierarchical algorithm was introduced by the same research group [37]. AHSCAN uses the same set of definitions as DHSCAN and the same core principle. In the beginning every separate vertex forms its own cluster. The algorithm sorts edges in descending order of edge structure value and removes them one by one, merging vertices (or clusters) connected by the chosen edge. After every merging, modularity (quality of clustering) is

recalculated and the highest value, as well as corresponding clustering, is saved. ASCAN does not require any input parameters, which is a definite advantage compared to SCAN. For some reason, the authors decided to use Newman's modularity in order to identify the best partition. Newman's modularity is 'well accepted by the research community' [37], but as the authors mentioned in previous publications [36] and [14], it failed to identify nodes of special importance: hubs and outliers. Thus AHSCAN in the form it is introduced by the authors does not explicitly identify hubs and outliers.

The authors claim their algorithm to be linear with respect to the number of edges, though complexity analysis is not provided in the paper, the result is just stated there. The authors also suggest that AHSCAN to their knowledge 'is one of the fastest, if not only, network clustering algorithms in literature' and the accuracy provided by the algorithm is claimed to be 'quite competitive with one of the state-of-art agglomerative network clustering algorithm, CNM, which has much higher complexity' [37]. CNM denotes Clauset-Newman Modularity, which in turn is denoted as Fast Modularity in this work.

AHSCAN pseudocode is provided below and is taken from [37]:

```
Algorithm AHSCAN(G = <V, E>)
CALCULATE all κ(e)          //  e ∈ E
SET all κ(e) into P[]
SORT P[] in descending order
i := SIZE(E)
Max_Q := 0;   // Max modularity
C:= {}            // Clusters at cut-off
```

```
FOR edge := 1 to i
MERGE vertices of P[edge]
CALCULATE Modularity
IF Modularity > Max_Q THEN
        Max_Q := Modularity
        STORE Clustering layout as C
ENDIF
ENDFOR
PRINT C
END AHSCAN.
```

Figure 3 - AHSCAN pseudocode

II.1.4. Weighted graphs

All the algorithms mentioned above have one common property - they target unweighted graphs. Every graph edge may have a positive number associated with it, which is usually called edge weight or capacity. Weight/capacity might have different interpretations from intensity of nodes interaction to the maximum amount of flow passing through the edge. In any case edge weight encapsulates some valuable information about the graph and the real-world structure it represents. When provided with a weighted graph, any of the algorithms mentioned above will simply ignore the edge weights and will perform clustering based on structural properties of the graph. While this might be acceptable in certain cases, sometimes it is completely inadmissible. Some networks (graphs) simply lose their meaning without weights: for instance a graph representing the network of

interconnected airports where edge weight corresponds to the distance between airports. The fact of the possible connection between airports is, of course, of a crucial importance, but without knowing the distance between them any operations on the graph would not make much sense. Thus it is logical to expect the value of the edge weight to influence the graph clustering.

Maximum flow-based algorithms may be considered an extension of minimum cut algorithms for weighted graphs. The value of maximum flow between any two nodes in graph is directly related to edge weights (or capacities). The cut definition for weighted graphs is the same, but it gains an additional property - the value. The sum of edge weights that constitute the cut is a cut value [16]. A cut with a minimum value is called a minimum cut. Ford and Fulkerson in [16] showed that maximum flow between two nodes is equal to the value of minimum cut, separating them. Based on this idea it is possible to build a minimum cut tree. A tree is an unweighted connected graph without cycles. Minimum cut tree of a graph G(V, E) is a tree that contains all the vertices of the graph G and weighted edges connect vertices in a way that a path between any two vertices has a capacity of a minimum cut.

One of well known maximum flow-based algorithms - minimum cut tree method proposed by Flake et al. in [15] - is based on calculation of the minimum cut tree of the graph using the algorithm proposed by Gomory and Hu in [18]. The clusters are achieved by extending the original graph with an artificial sink *t* and connecting every existing vertex to it by an edge with weight α. Using Gomory and Hu's algorithm the min cut tree is then obtained. Removing the artificial node from the tree and getting the connected components produce the connected elements that form achieved clusters.

Minimum cut tree clustering algorithm targets undirected weighted graphs. Being a maximum flow based method it takes edge weights into consideration. Edge weights have direct influence on the flow value, which in turn is used by clustering algorithm. The algorithm requires an input parameter: the weight α of additional edges that connect all the vertices to the artificial sink. Flake et al. analyze the influence of α value on the clustering: when α is close to zero, the trivial cut, separating sink *t* from all other nodes in the graph, will be minimum. If α in turn goes to infinity, the min cut tree turns into a star, thus removing of a sink produces n singleton clusters, where n is the number of nodes in the graph. In that way selection of α is important for the clustering result. Thus the algorithm is capable of identifying isolated nodes, but it happens more as a by-product of choosing inappropriate parameter value, so it is not likely that isolated vertices will be the ones of some special role in the graph. In that way, mentioned algorithm has the same drawbacks as most of methods mentioned before: inability to identify special nodes, presence of input parameter, that might be hard for the user to determine.

It is worth mentioning that Fast Modularity algorithm, mentioned before as one of the top clustering algorithms, originally targeting unweighted graphs, can be easily extended to utilize edge weights. This procedure is described by Newman in [25].

II.2. Quality measurements

There are few ways of checking the quality of the achieved graph partition. When actual partition is known a priory (if there exist known natural clusters in a form of divisions, teams, groups, etc), achieved clustering can be compared to real clusters and thus quality of clustering, produced by an algorithm, can be estimated [20]. Unfortunately natural

clustering, if existent, is rarely known a priory and consequently other methods should be used to estimate the results. Quality assessment measures (functions) is another approach. Such function takes graph partitioning (clustering) as input and calculates a value, usually ranging from 0 to 1, that indicates the quality of partitioning. The higher the value, the better the quality of the clustering.

Brandes and Erlebach in [2] describe the rules for such measurement design. The process of clustering usually focuses on two main properties: intra-cluster density and inter-cluster sparsity. Thus a good quality measurement function should take them both into consideration. Denoting density inside clusters with f and sparsity between clusters with g, Brandes and Erlebach define the general look of such function by the formula:

$$index(C) = \frac{f(C)+g(C)}{\max\{f(C')+g(C'):C' \in A(G)\}} \quad (9)$$

where A(G) is the set of all possible partitions. In this way the function achieves its maximum value of 1 for the 'extremely' fitting clustering and the minimum value of 0 in case of absence of meaningful clusters or totally random.

II.2.1. Newman's modularity

The most popular and well accepted by the researchers quality measurement is modularity proposed by Newman and Girvan in [27]. To calculate the modularity one needs to construct symmetric *k* by *k* matrix *e*, where *k* is number of clusters. Diagonal elements e_{ii} of the matrix contain the fraction of graph edges that connect vertices in *i*-th cluster, while the rest matrix elements e_{ij} contain the fraction of edges connecting *i*-th cluster to *j*-th one. The trace of the matrix:

$$Tr\ e = \sum_i e_{ii} \tag{10}$$

produces the fraction of intra-cluster edges of the graph. Matrix trace corresponds to density *f* in Brandes and Erlebach work [2] and is not sufficient for adequate clustering assessment by itself, as a single cluster containing all graph vertices will result in maximum trace value equal to 1 [27]. To include sparsity g in the estimate the 'row (or column) sums' are introduced:

$$a_i = \sum_j e_{ij} \tag{11}$$

that represent the fraction of edges connecting to the vertices in cluster *i*. Thus the authors define modularity:

$$Q = \sum_i \left(e_{ij} - a_i^2\right) = Tr\ e - \|e^2\| \tag{12}$$

where $\|e\|$ denotes the sum of the elements of the matrix *e* [27].

II.2.2. Novel Similarity-Based Modularity

Feng et al. in [14] criticize Newman's modularity. The authors claim that Newman's modularity works very well with clear clustering structure, where the inter-cluster connections are sparse. But modularity method fails to process effectively partitions with 'hub' and 'outlier' types of vertices that do not belong to any particular cluster or produce singleton clusters. To address this issue the authors propose similarity-based modularity for the graph V:

$$Q_s = \sum_{i=1}^{NC} \left(\frac{IS_i}{TS} - \left(\frac{DS_i}{TS}\right)^2\right) \tag{13}$$

where

$$IS_i = \sum_{u,v \in V_i} \sigma(u, v), \tag{14}$$

$$DS_i = \sum_{u \in V_i, v \in V} \sigma(u, v), \quad (15)$$

$$TS = \sum_{u,v \in V} \sigma(u, v), \quad (16)$$

NC is number of clusters, IS_i - total similarity of vertices in i-th cluster, DS_i- total similarity between any vertices in the graph and vertices in cluster i, TS - total similarity between any two vertices in the graph; $\sigma(u, v)$- structural similarity of vertices u and v, defined before.

Feng et al. claim that their similarity-based modularity produces better results in graphs with 'more confused' sub-graph structure, including vertices of special importance: hubs and outliers [14].

II.2.3. Performance

Performance is another quality measure that combines measures for inter-cluster sparsity *g* and intra-cluster density *f*. Performance was first proposed by Iverson in [21] and later adapted by Knuth in [23]. Performance counts particular node pairs. As for the density *f*, it consists of 'correctly' classified pairs that belong to the same cluster and are connected by an edge; the other case, intra-cluster sparsity *g* consists of 'nonexistent edges' - pairs that belong to distinct clusters and are not connected by an edge - [2]:

$$f(C) := \sum_{i=1}^{k} |E(C_i)| \quad (17)$$

$$g(C) := \sum_{u,v \in V} [(u, v) \notin E] \cdot \left[u \in C_i, v \in C_j, i \neq j\right] \quad (18)$$

where *C* is clustering of the graph *G(V, E)*, $E(C)$ - the set of intra-cluster edges.

Calculating maximum of *f* + *g* is NP-hard [29], but obviously is has *n(n - 1)* as the upper bound (where n is the number of nodes), simply because there are *n(n - 1)* possible node pairs.

Brandes and Erlebach in [2] provide the derivation of the formula for performance calculation:

$$perf(C) = 1 - \frac{m\left(1-2\frac{m(C)}{m}\right)+\sum_{i=1}^{k}|C_i|(|C_i|-1)}{n(n-1)} \quad (19)$$

where m is the number of edges in the graph, and $m(C)$ is the number of intra-cluster edges in the partition.

II.3. Graph Generators

Random or generated graphs are often used in different experiments that involve graphs and graph clustering is not an exception. Real world datasets might be hard to access and/or require significant effort to parse and transform to graph form. Therefore graph generators provide relatively easy solution. Properly generated graph will have (or at least will be expected to have) predefined community structure and thus may be used to validate clustering algorithms. Also many years of study reveal a fact that real world graphs follow certain patterns that keep reoccurring in various fields of science. Surveys like the one by Costa et al. [7] list these patterns.

II.3.1. Planted l-partition model

A famous graph-generating framework was proposed by Girvan and Newman and originally used in [17], being applied afterwards in many research papers: [27], [26] and others: [19]. This framework is a special case of so-called l-partitioned model originally proposed by Condon and Karp in [6]. L-partitioned model has predefined community structure, thus it can be used to validate the graph-clustering algorithms. A graph in the

model has $n = g * l$ vertices, where l is number of clusters with g vertices in each. Vertices are connected with probabilities p_{in} for intra-cluster edges and p_{out} for inter-cluster ones. The average degree can be calculated using the following formula:

$$k = p_{in}(g-1) + p_{out}g(l-1) \quad (20)$$

Community structure in the graph is present when $p_{in} > p_{out}$ (which means that intra-cluster density is higher than inter-cluster density).

Girvan and Newman framework is a special case of planted l-partition model with $l = 4$ and $g = 32$ (number of vertices in graph is 128) and average degree k equal to 16. The graph generator has a parameter that specifies the level of intra-cluster density: the average number of inter-cluster edges z_{out} for every vertex (vertex outer degree). Thus the distinguishable community structure can be obtained for z_{out} values up to 8. Various types of community structure are depicted on the Figures 4 to 6:

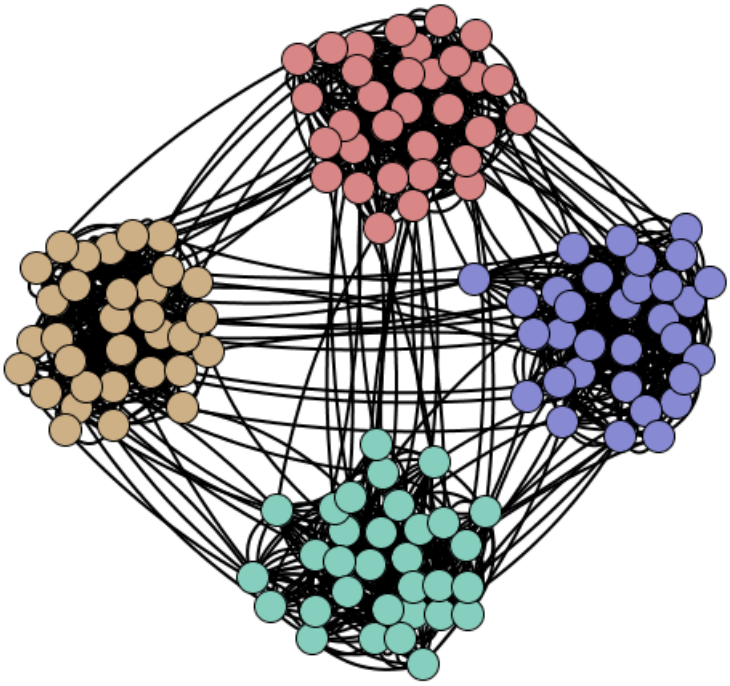

Figure 4 - Community structure with 1 inter-cluster edge per node

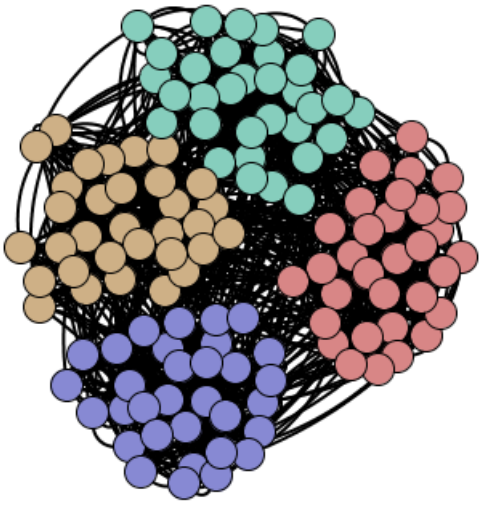

Figure 5 - Community structure with 4 inter-cluster edges per node

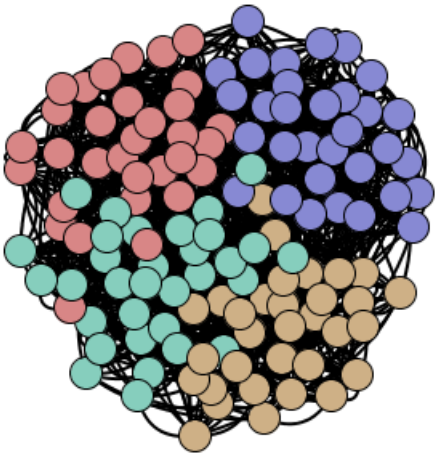

Figure 6 - Community structure with 6 inter-cluster edges per node

Normally it is hardly possible to identify community structure with z_{out} more or equal to 8.

II.3.2. Preferential attachment

In [1] Barabasi, studying real-world networks (like genetic networks or World Wide Web) discovered an effect of "preferential attachment" - when the network expands by

gaining new vertices, they (new vertices) are more likely to become connected to a high degree vertex. Barabasi proposed a method for graph generation that begins having a small number of vertices and adds a new one (as well as specified number of edges) at every time step probability. Edges are attached to existing vertices in order to incorporate preferential attachment principle: the probability to be connected to the existing vertex *i* is:

$$p = \frac{degree(v)}{|E|} \tag{21}$$

where $|E|$ is the number of existing edges in the graph.

The developers of an open source JUNG graph library implemented this graph generating method and included it in their library with some modifications. They noticed that the formula x always produces zero probability for an isolated vertex. Thus to "give each existing vertex a positive attachment probability" developers utilized the following formula:

$$p = \frac{degree(v)+1}{|E|+|V|} \tag{22}$$

where $|V|$ is the number of vertices already existent in the graph.

II.3.3. Power laws

Another important feature found for web-graphs in the same work [1] by Barabasi and Albert is power-law degree distribution, which happens to be a consequence of preferential attachment and incremental growth. Due to Barabasi and Albert power law degree distribution consists in the fact that probability of a vertex in the graph with a degree equal to *k* is proportional to $k^{-\gamma}$:

$$P(k) \sim k^{-\gamma} \quad (23)$$

Eppstein and Wang in [11] suspect power law degree distribution to be an ubiquitous property of real-world graphs. At least the authors state that it is present in epidemiology, population studies, genome distribution, etc. The authors propose a graph generation method that does not require incremental growth, instead it utilizes and evolve an existing graph according to a Markov process. The iterative algorithm is provided below:

1. Pick a random vertex v with non-zero degree.

2. Pick an edge (u, v) from the graph at random

3. Pick another random vertex x.

4. Pick a vertex y not equal to x with a proportional to degree probability.

5. If an edge (x, y) does not exist in the graph, remove the edge (u, v) and add edge (x, y).

II.3.4. Community structure

Defining community as a subgraph where vertices are closer to each other (in terms of similarity functions) than to the ones in other subgraphs, the fact that community structures are ubiquitous in real world graphs probably can be considered indirectly proved by variety of clustering algorithms existent today. In social networks community structure reveals itself by a simple pattern: individuals usually have more connections to other members of the same community than to ones from other groups [22]. A number of metrics to measure the clumpiness of a graph was proposed. One of the most basic and common ways is a clustering coefficient. Clustering coefficient is most often defined for

the graph [24], [3], but there also exists an alternative definition of a clustering coefficient of a single vertex provided by Watts and Strogatz in [33]. The latter is of a particular interest for us, thus we will explain it here in more detail.

Clustering coefficient for a vertex *i* is defined as

$$C_i = \begin{cases} \frac{n_i}{k_i}, & k > 1 \\ 0, \; k_i = 0 \; or \; 1 \end{cases} \tag{24}$$

where k_i is the number of vertex *i* neighbours and n_i is the number of edges between the neighbours. The central node X on Figure 7 has 6 neighbours, and there are 5 edges between the neighbours (not including the edges incident to X). Thus local clustering coefficient of the node X is 5/6.

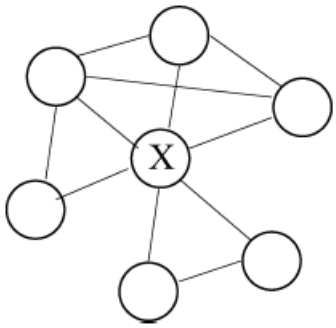

Figure 7 - Clustering coefficient

CHAPTER III

DESIGN AND METHODOLOGY

III.1. Extended structural similarity

Among the variety of existing clustering algorithms SCAN and other based on it algorithms (DHSCAN and AHSCAN) distinguish themselves as they exhibit some important features: the ability to identify nodes of a special interest - hubs and outliers, and to a greater extent - focus on neighbourhood for identifying clusters. This 'social' aspect, as all three algorithms are based on the structural similarity - the value derived from the neighbourhood structure, the idea of using information about neighbours to make a decision about clustering, makes these algorithms particularly promising for the sake of Social Network Analysis. Intuitively this property seems to have a great validity in real world social networks. In the famous experiment by Travers and Milgram in [32] people had to send a chain letter to reach a random person of interest. Thus people used only their 'direct neighbourhood' - only people they knew - to send a letter intended to someone they do not know. The experiment revealed the concept of six degrees of separation: the average length of chain was six, which is considered a very small number taking into the account the size of population affected. This experiment shows the importance of the neighbours in the social graph. Thus the idea of using this information for clustering seems to us extremely valuable.

No other known clustering algorithms possess both of these features: usage of neighbourhood information and ability to identify nodes of special importance. On the other hand, a serious limitation of SCAN based algorithms is the fact that they target unweighted graphs. The importance of edge weights in graphs is already mentioned in

chapter II. To avoid this limitation we introduce the weighted versions of SCAN and SCAN-based algorithms. These algorithms use modified structural similarity that incorporates edge weights.

In [4] a weighted version of SCAN which allows to overcome the mentioned limitations was already introduced. However, the new algorithm was just presented, and was never compared to any other existing clustering algorithm. Moreover the extension created another drawback. The proposed algorithm used slightly modified formula for structural similarity:

$$\sigma_{ex}(\vartheta,\omega) = e(\vartheta,\omega)\frac{|\Gamma(\vartheta)\cap\Gamma(\omega)|}{\sqrt{|\Gamma(\vartheta)||\Gamma(\omega)|}} \tag{25}$$

where $e(\vartheta,\omega)$ is a weight of the edge connecting vertices ϑ and ω.

Adopting edge weights, this formula at the same time introduces a limitation: similarity between vertices that are not connected by an edge turns to zero, which significantly limits the potential of the mentioned social structure concept based on neighbourhood.

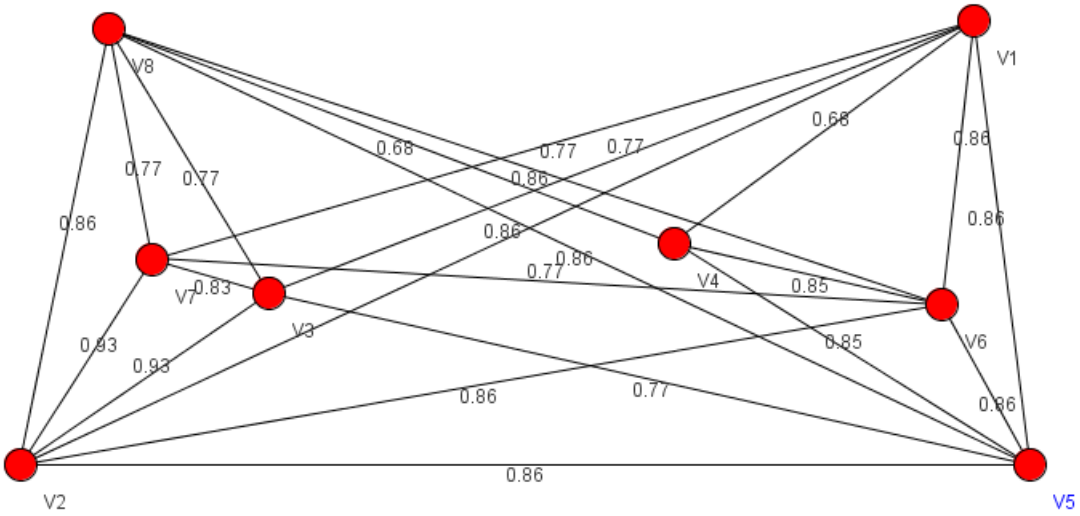

Figure 8 - numbers under the edges correspond to the structural similarity of an edge.

Indeed vertices V1 and V8 on figure 4 are not connected on the picture, but if we calculate the structural similarity between them, we will get rather high value of 0.857, which would normally mean that these two vertices are very similar.

In this work we propose modifications to Weighted SCAN in order to improve its ability to detect clusters and apply the same approach to derived DHSCAN and ASCAN.

To overcome the limitations of Chertov's approach we introduce extended structural similarity defined like this:

$$\sigma_{ex}(\vartheta,\omega) = \left(1 + \frac{e(\vartheta,\omega)}{e_{max}}\right)\frac{|\Gamma(\vartheta)\cap\Gamma(\omega)|}{\sqrt{|\Gamma(\vartheta)||\Gamma(\omega)|}} \quad (26)$$

where e_{max} is the maximum weight in the graph or

$$\sigma_{ex}(\vartheta,\omega) = (1 + e_{norm}(\vartheta,\omega))\frac{|\Gamma(\vartheta)\cap\Gamma(\omega)|}{\sqrt{|\Gamma(\vartheta)||\Gamma(\omega)|}} \quad (27)$$

where $e_{norm}(\vartheta,\omega)$ is normalized weight of the edge connecting vertices ϑ and ω. Thus a graph needs to be 'normalized' before weighted SCAN can be applied to its edge weight values.

The base of the similarity between two vertices remains the same - similarity of people sharing many friends, but not knowing each other (not being connected by an edge in the graph) remains the same, while the existence of the connection (an edge) between them only increases the value of structural similarity.

Thus all the algorithms (SCAN and derived from it DHSCAN and ASCAN) remain the same, except the formula of structural similarity used at many steps of the algorithms. The change in the absolute value of the similarity affects SCAN, but none of its derivatives as they use comparative value of similarity and similarity of almost every pair of vertices is affected. In case of modified version of SCAN adjustment of ε-parameter (in comparison to unweighted algorithm) is required.

Since now we will not go back to unweighted versions of the algorithms and by SCAN, DHSCAN and AHSCAN we will denote weighted versions of these algorithms, which use the aforementioned approach (formulas 26 and 27) to structural similarity (unless mentioned explicitly).

III.2. Clustering quality assessment

For quality assessment of the clustering result, we use adapted Newman's modularity and Similarity-Based Modularity, mentioned in chapter II.

III.2.1. Newman's modularity

Modularity, introduced by Newman and Girvan in [27] targets unweighted graphs. Matrix e, used in calculations, is populated with regard to the number (and fraction) of edges connecting vertices in the graph, vertices in the same cluster and vertices in different clusters. In [25] Newman describes the idea of its extension: using actual edge weights instead of counting the number of edges. The same formula for modularity calculation is used:

$$Q = \sum_i \left(e_{ij} - a^2\right) = Tr\ e - \|e^2\| \quad (28)$$

Instead of number of edges in the graph now the total 'graph weight' is calculated - the sum of all edge weights in the graph. For diagonal elements of matrix *e* (e_{ii}) the sum of edge weights connecting vertices in cluster *i* and then find the fraction of this sum weight in the 'graph weight' is computed. Similarly for regular elements e_{ij} the fraction of edge weights connecting vertices in cluster *i* to vertices in cluster *j* is calculated.

The derived modularity formula as defined in [25] is given below:

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j) \quad (29)$$

where A is adjacency matrix of the graph, δ function is defined like:

$$\delta(u, v) = \begin{cases} 1, & if\ u = v, \\ 0, & otherwise \end{cases} \quad (30)$$

m is the number of edges in the graph, k_i - is the degree of vertex *i*.

For weighted networks, matrix *A* turns into an extended adjacency matrix, where A_{ij} corresponds to the weight of the edge connecting vertices *i* and *j*, degree k_i is defined according to the formulas:

$$k_i = \sum_j A_{ij} \quad (31)$$

$$m = \frac{1}{2}\sum_{ij} A_{ij} \quad (32)$$

Non zero values of modularity 'indicate deviations from randomness', while values around 0.3 or more usually correspond to good partitions. Maximum possible modularity value is 1 [25].

III.2.2. Similarity-based modularity

Similarly to Newman's modularity similarity-based modularity was proposed to measure the quality of unweighted graph clustering. As mentioned in chapter II, the value of similarity-based modularity is calculated based on structural similarity. SCAN, DHSCAN and AHSCAN are also based on the notion of structural similarity. Analogously to extending the clustering algorithms by making them use weighted structural similarity, we substitute similarity-based modularity with its extended weighted version:

$$Q_s = \sum_{i=1}^{NC}\left(\frac{IS_i}{TS} - \left(\frac{DS_i}{TS}\right)^2\right) \quad (33)$$

where

$$IS_i = \sum_{u,v \in V_i} \sigma_{ex}(u, v), \tag{34}$$

$$DS_i = \sum_{u \in V_i, v \in V} \sigma_{ex}(u, v), \tag{35}$$

$$TS = \sum_{u,v \in V} \sigma_{ex}(u, v), \tag{36}$$

NC is number of clusters, IS_i - total extended similarity of vertices in i-th cluster, DS_i- total extended similarity between any vertices in the graph and vertices in cluster i, TS - total extended similarity between any two vertices in the graph; $\sigma_{ex}(u, v)$ - extended structural similarity of vertices u and v, defined before.

The table 1 provides the comparison of the values of the unweighted and weighted Q_s for different partitions of the graph of the figure 10.

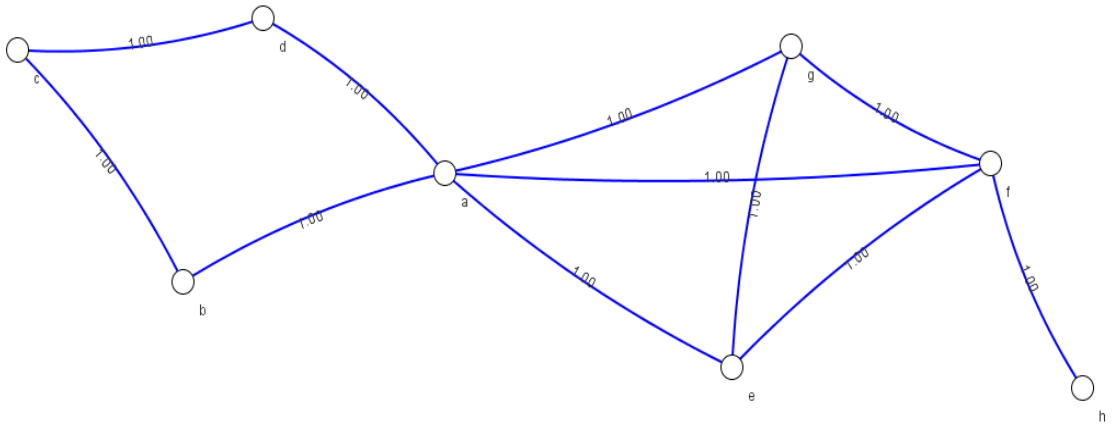

Figure 9 - Sample graph with plain weight distribution

Clustering structure	Unweighted Qs	Weighted Qs
All vertices in one cluster	0	0
{a }, {b, c, d}, {f, e, g}, {h}	0.22727	0.22052
{a, b, c, d}, {e, f, g}, {h}	0.21491	0.21091
{ a b c d }{ e f g h }	0.22650	0.22415
{f e a g }{h }{b d c }	0.23582	0.23591
{b, c, d},{a, e, f, g, h}	0.23916	0.23985
{a},{b, c, d},{e, f, g, h}	0.23886	0.23376

Table 1 - Weighted and unweighted structural similarity-based modularity.

These values are a little different from the results published by Feng et al. in [14] where the similarity-based modularity was first introduced. Feng et al. had the maximum modularity value for {a},{b, c, d},{e, f, g, h}, while {b, c, d},{a, e, f, g, h} had a little less. We have the opposite situation, though the values are very close to each other. This gives rise to questioning the validity of similarity-based modularity. After contacting the authors we resolved ambiguity in modularity calculation and multiple checks of the calculation algorithm did not identify any reason for us having different values of the quality functions. However, introducing weights into the computation does not significantly change the relative function values. The best identified partition is still the same.

If we increase the weight of one single edge *fh* to make it five times stronger than all other edges (to keep the values in [0, 1] range we also decrease values of other edges), we will be able to notice some changes in the weighted similarity-based modularity value.

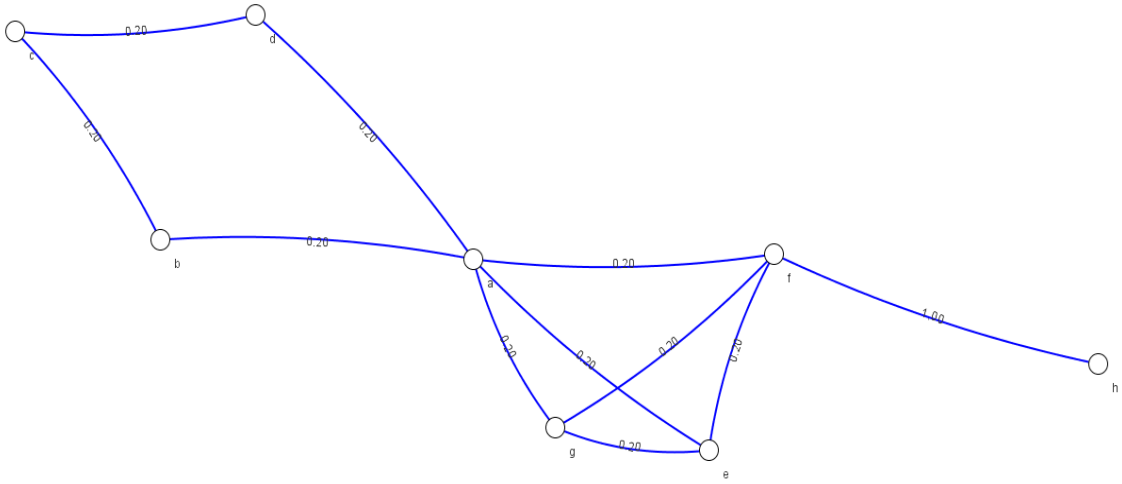

Figure 10 - Graph with different weight distribution

The weight of an outlier edge connecting *g* and *f* is 5 times bigger - we see tendency to put *h* into an adjacent cluster.

Clustering structure	Q unweighted	Weighted Q
All vertices in one cluster	0	0
{a }, {b, c, d}, {f, e, g}, {h}	0.22727	0.20852
{a, b, c, d}, {e, f, g}, {h}	0.21491	0.20195
{ a b c d }{ e f g h }	0.22650	0.23008
{f e a g }{h }{b d c }	0.23582	0.22302
{b d c }{f e a g h }	0.23916	0.23833
{a }{b d c }{f e g h }	0.23886	0.17328

Table 2 - Weighted and unweighted similarity-based modularity values

The observed behaviour can be considered rather strange, but we can still see the tendency to identify hubs and outliers: the absolute value of the partitions with isolated vertex *h* went noticeably down.

It is worth to be noted again, that DHSCAN and AHSCAN, being developed by the same group of scholars and having same principle in their core, use different

clustering quality measurements to find an optimal solution. It might imply that similarity-based modularity is not as good as it was thought to be originally. We are going to check it in our experiments.

III.2.3. Performance

For weighted performance we will use one of the variations included in [2], where $f(C)$ is sum of weights of intra-cluster edges:

$$f(C) := \sum_{i=1}^{k} w(E(C_i)) \tag{37}$$

and $g(C)$ is maximum possible weight of non-existent inter-cluster edges:

$$g(C) := \sum_{u,v \in V} M \cdot [(u,v) \notin E] \cdot \left[u \in C_i, v \in C_j, i \neq j\right] \tag{38}$$

where C is clustering of the graph G(V, E), and M is some meaningful maximum edge weight.

The value of *M* may introduce some disturbance into calculation, if weights are not uniformly distributed - in other words the presence of a single extremely large edge weight will influence the performance significantly, disrupting 'the range aspects' of the quality measurement [2].

The obvious drawback of this approach is neglecting the weight of inter-cluster edges. The following modification can be used:

$$g'(C) = g(C) + g_w(C) \tag{39}$$

where

$$g_w(C) = M \cdot \left|\overline{E(C)}\right| - w\left|\overline{E(C)}\right| \tag{40}$$

and $\overline{E(C)}$ is a set of inter-cluster edges in partition C.

The final formula will look like this:

$$perf_w(C) = \frac{f(C)+g(C)+\nu \cdot g_w(C)}{n(n-1)M} \quad (41)$$

where $\nu \in [0,1]$ is a parameter that scales the importance of the inter-cluster edge weights.

The meaningful maximum edge weight *M* will be equal to 1 for our study. It was already mentioned that extended structural similarity uses normalized value of edge weight, and for the sake of easiness of experiments implementation, our graphs will be always normalized.

III.3. Graph generators

III.3.1. Planted l-partition model

In the work devoted to the analysis of weighted networks [25] Newman describes the way to extend their framework and utilize edge weights. The idea behind this approach is simple and consists of allocating bigger weights for the intra-cluster edges than for the inter-cluster ones. We also extend this framework adding artificial hubs and outliers into the network (total of 4 special nodes), as we expect proposed algorithms to be able to identify them. The expected clustering picture is shown on the figure below. White nodes are hubs, black - outliers:

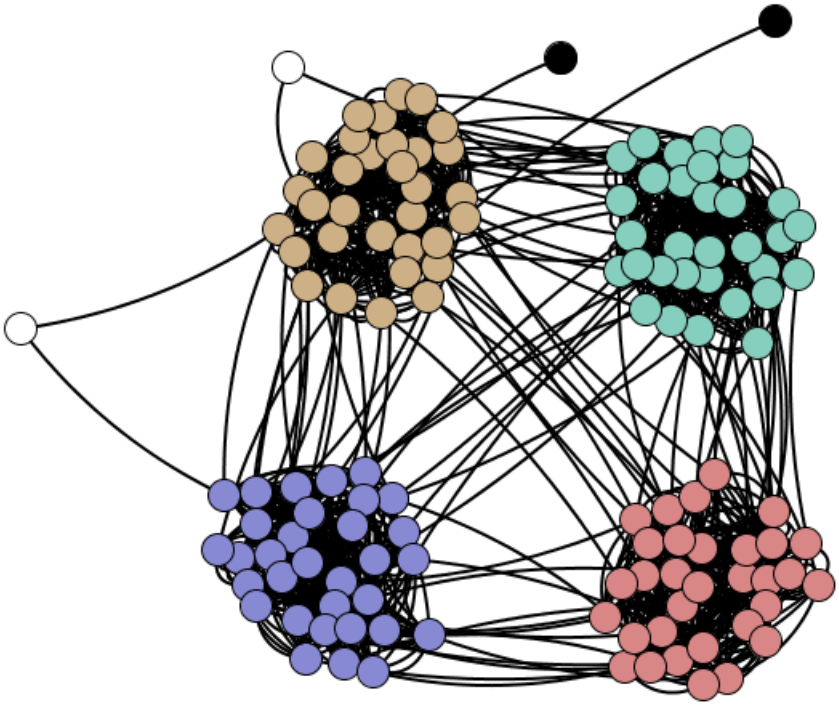

Figure 11 - Synthetic graph with hubs and outliers

III.3.2. Other types of graphs

As data sets in the experiments we are going to use graph generated by methods described in chapter II. But they also have a limitation - they produce unweighted graphs. The meaning of weights in most real-world networks is some sort of similarity between vertices. Thus vertices connected by an edge with a high weight value are similar and have higher probability of being in one community. To incorporate this notion we decided to calculate clustering coefficient, described in chapter II and use these values to assign edge weights. For an edge connecting vertices *u* and *v* the weight value is calculated using the following formula:

$$w(u, v) = max\ (clusteringCoef(u), clusteringCoef(v)) \quad (42)$$

This approach populates the edges with meaningful weights, but the direct relation of the

weight values to the graph structure may be considered a significant drawback at the same time.

III.4. Complexity analysis

Complexity analysis of SCAN was provided before in chapter II. Complexity of weighted version of the algorithm is not expected to change. While there are three extra arithmetic operations in vertices similarity calculation, the algorithm is not changed: it is still required to pass every edge twice (once from each end), which ends up complexity being $O(m)$, where m is number of edges.

Similarly hierarchical algorithms (DHSCAN and AHSCAN) still perform the same sequence and number of operations: DHSCAN requires $m - 1$ edge removals, AHSCAN requires $n - 1$ vertex agglomerations, requiring only 3 extra arithmetic operations for every edge similarity value.

III.5. Approach

The aim of this research is to prove (or disprove) the validity of our approach to extension of SCAN based algorithms. For that purpose we will run a set of experiments on real and generated graphs.

A number of pretests is usually run to define the best value for SCAN ε input parameter. Weighted Fast Modularity algorithm is used as a reference clustering method. New algorithms results are compared to the results produced by the reference algorithm as well as to random clustering results with the help of quality functions. Three quality measures are calculated and compared for every partition produced by every algorithm. We also implemented random graphs clusterer, producing random partitions to provide 'ground level' for quality measurement comparison.

CHAPTER IV

ANALYSIS OF RESULTS

IV.1. Generated (random) graphs

IV.1.1. Planted l-partition model

Planted l-partition model described earlier has an input parameter: z_{out}, which corresponds to the number of inter-cluster edges per vertex. Community structure becomes less obvious with the increase of z_{out}, which produces a challenge for clustering methods. The number of clusters is 4, average vertex degree is 16. Thus for $z_{out} = 8$ no identifiable community structure is expected to exist. Most of the experiment results are shown in the diagrams. Every point on the diagram corresponds to the average of 100 random graphs.

IV.1.1.1. DHSCAN and AHSCAN

Before we apply weighted modifications and move to the analysis of the cumulative algorithmic charts we would like to pay some attention to hierarchical SCAN-based algorithms. As clustering quality measurement DHSCAN uses similarity-based modularity, which was proposed by the same research group in [14] and is claimed to be better at identifying special nodes in graphs. Thus the paper, introducing DHSCAN [36], suggests that the algorithm is capable of identifying hubs and outliers, but experiment section does not provide any information about detecting special nodes. The paper also does not contain the results of experiments against planted 4-partition model benchmark. Our analysis of similarity-based modularity, provided earlier in chapter III, discovered that even though this quality measurement accommodates hubs and outliers in partitions

better, it does not provide 100% accuracy. Thus the question on how good DHSCAN is at detecting special nodes is of an interest here. For that reason we decided to check the ability of original unweighted DHSCAN to identify hubs and outliers. We ran a number of experiments using planted 4-partition model. The chart below shows the results:

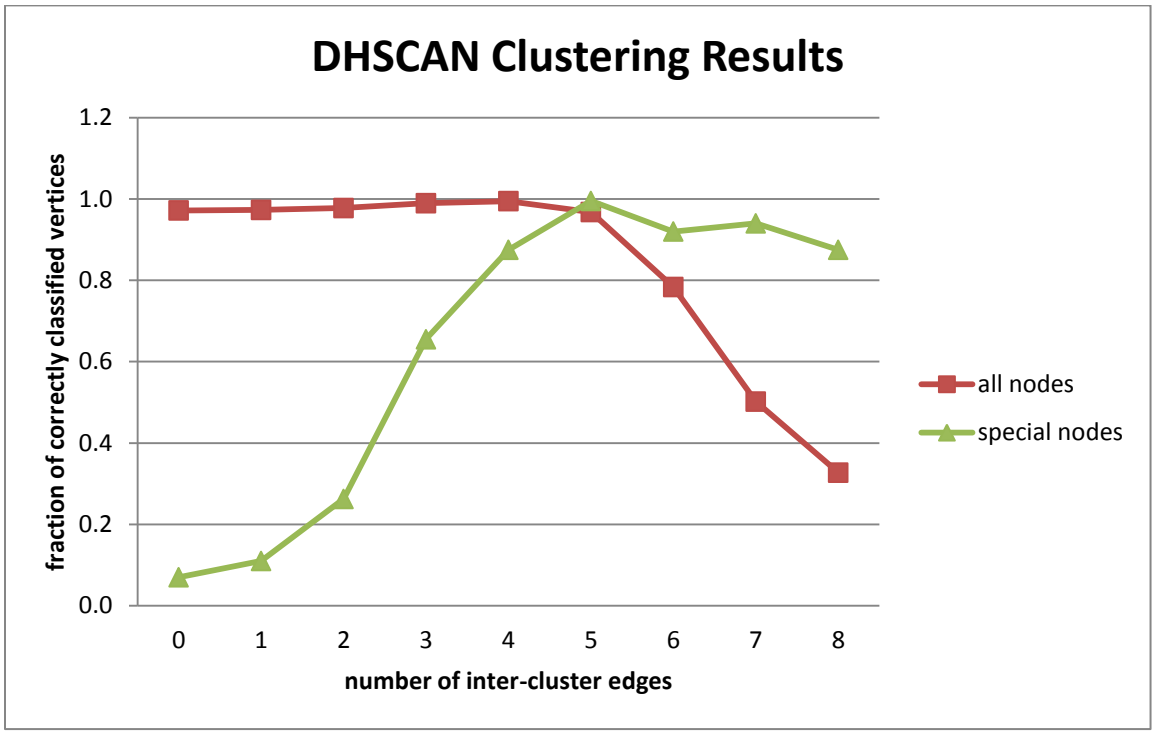

Figure 12 - Regular (unweighted) DHSCAN detecting special nodes. Every point of the chart corresponds to the average value of 100 random graphs.

Discovered behaviour is rather strange: DHSCAN detects special nodes only in certain range of z_{out} values, which is hard to determine. Also one should be careful with interpreting good results for z_{out} equal to 7 and 8. We were able to identify that with the increase of inter-cluster communication clustering algorithm produces more singleton clusters (and less vertices are classified correctly overall). Whenever special node happens to be in such singleton cluster is it considered correctly classified. As we will see

later, DHSCAN is still better than Fast Modularity, but obviously worse than SCAN in detecting special nodes.

It is worth noting that AHSCAN, proposed by the same research group two years after and based on the same principles as DHSCAN, does not use similarity-based modularity to identify the best partition, but takes use of classical Newman's modularity and thus is not claimed to be able to detect special nodes. That makes it possible to surmise that proposed hierarchical approach is not suitable for effective detecting of special nodes.

We decided to tweak AHSCAN to use similarity-based modularity and run the same set of experiments using this new version of AHSCAN. We do not provide the chart as it looks almost identical to the one on figure 12.

IV.1.1.2. Overall picture

In this section we provide cumulative algorithmic charts. Every random generated graph was clustered by every studied algorithm.

The characteristic feature and the main drawback of SCAN is the presence of two input parameters: ε and μ. The authors of the algorithm advise to always use 2 as the value for μ (which is the minimal cluster size) [35], but the optimal value of ε might be different for various graphs. To identify the best ε-value we did a set of preliminary experiments with random graphs generated using planted l-partition model. The value of ε that produces the best partitions for planted l-partition model graphs were found to be 0.6 for this type of graphs. Hierarchical SCAN-based algorithms do not require input parameters. We run experiments over two versions of AHSCAN: the original one, using

the Newman's modularity as quality measurement (denoted as AHSCANn), and the tweaked one, which uses similarity-based modularity instead (denoted AHSCANs). We also tried different edge weights distributions to see the influence that weights have on the clustering. We provide the graphics and descriptions below.

At first we try weighted algorithms on weighted graphs with plain weight distribution: all weights are the same.

Weights of intra-cluster edges	0.5
Weights of inter-cluster edges	0.5
Weights of the hub-incident edges	0.5
Weights of the outlier-incident edges	0.5

Table 3 - Graph parameters. Experiment 1.

Figure 13 - Fraction of vertices classified correctly. Every point of the chart corresponds to the average value of 100 random graphs.

Figure 13 shows that weighted SCAN-based algorithms perform noticeably better than weighted Fast Modularity, classifying more vertices correctly. Hierarchical algorithms produce almost identical results.

Every point on every consequent chart in this section corresponds to the average of 100 random graphs. The average standard deviations of the values of our interest for a typical experiment run are provided in table 4.

	FM	SCAN	DHSCAN	AHSCANs	AHSCANn
Node ratio	0.04	0.035	0.025	0.026	0.025
SN ratio	0.26	0.1	0.24	0.23	0.17
Modularity	0.027	0.011	0.011	0.014	0.002
SB Modularity	0.026	0.039	0.007	0.007	0.004
Performance	0.004	0.03	0.005	0.006	0.006

Table 4 - Standard deviation values

Results are rather consistent. Relatively large values for special nodes ratio is explained by small number of special nodes in the graph: there are only 4 special nodes in every graph (2 hubs and 2 outliers), thus misclassification of only one vertex reduces the ratio by 25%.

The following chart shows how the algorithms detected special nodes.

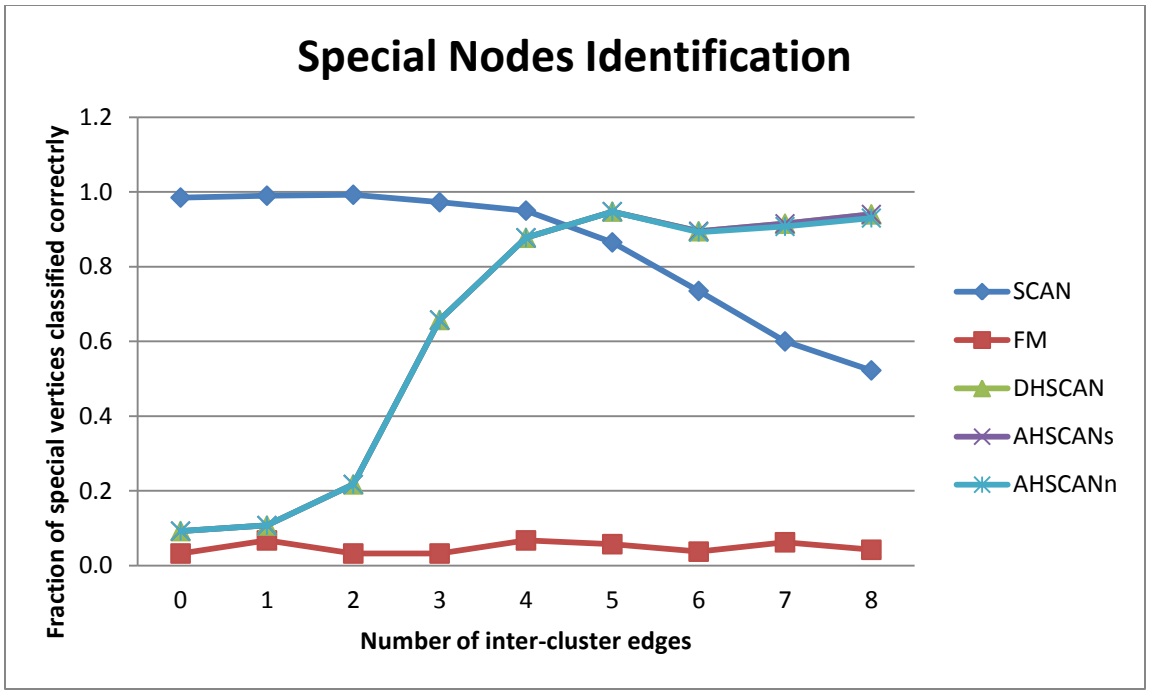

Figure 14 - Detection of special nodes by various algorithms. Every point of the chart corresponds to the average value of 100 random graphs.

The inability of Fast Modularity algorithm to detect nodes of special importance is obvious. We can also see that hierarchical algorithms produce almost identical results. Divisive or agglomerative nature as well as the quality measurement do not make any difference for this type of graphs. As following experiments will show, this is often the case. Since now we will leave only one diagram to represent all three hierarchical algorithms whenever produced results are similar.

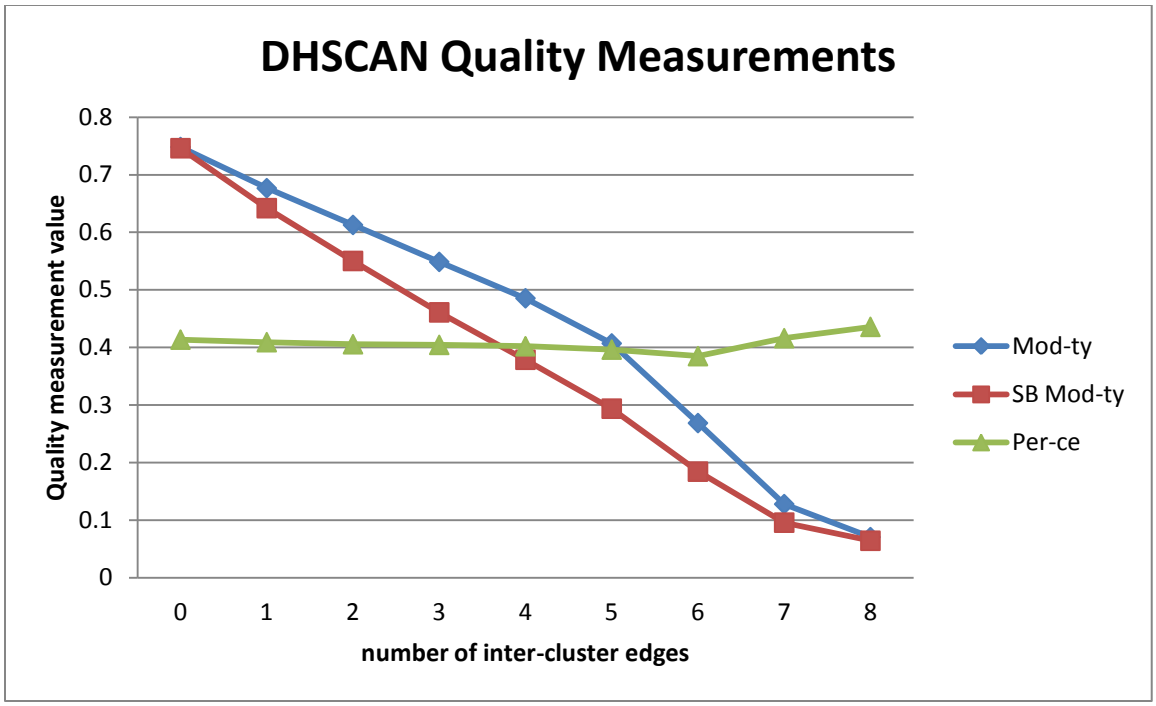

Figure 15 - Different quality measurements of DHSCAN clustering results. Every point of the chart corresponds to the average value of 100 random graphs.

Figure 15 depicts diagrams for different quality measurements of the partitions produced by weighted DHSCAN algorithm: modularity, similarity-based modularity and performance. We do not provide such diagrams for other algorithms because they are very similar to figure 15. We can see that performance is gradually and barely noticeably decreases with the reduction of the fraction of correctly classified nodes. The growth of performance, which starts with $z_{out} = 6$ can be explained by the increase of singleton clusters number. Singleton clusters usually contribute to performance with the non-existent inter-cluster edges, that compound inter-cluster sparsity. Modularity and similarity-based modularity seem to correlate and not to conflict with each other.

We also provide comparison of partitions quality, which are in sync with the number of correctly classified nodes. Figure 16 shows modularity values for partitions

produced by different algorithms. Values for all hierarchical SCAN-based methods were identical, so we denoted them all with HSCAN on the chart.

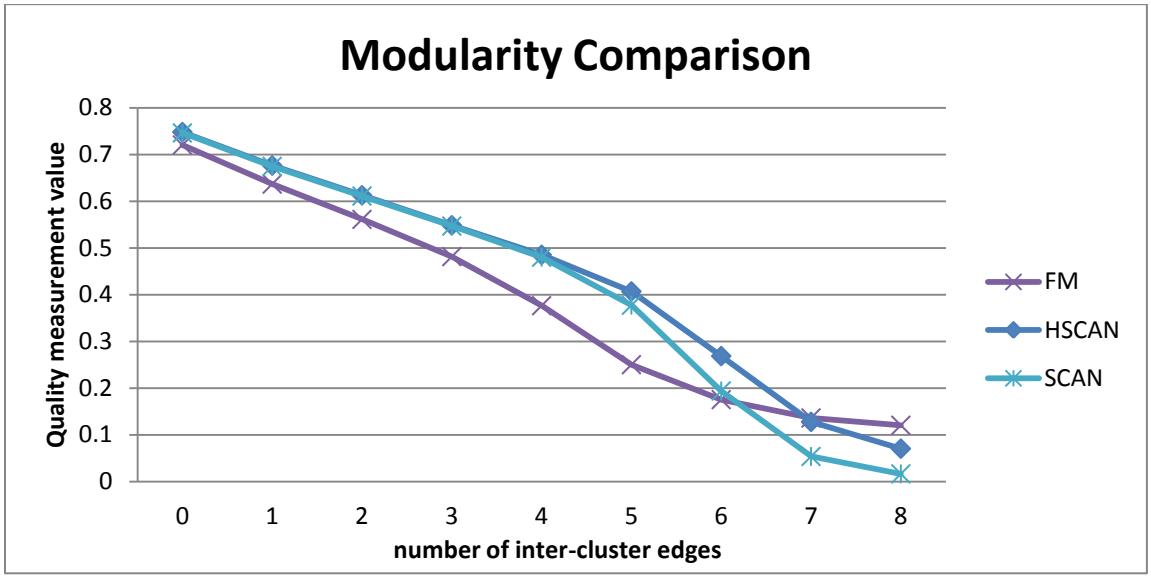

Figure 16 - Modularity values for partitions, produced by different algorithms. Abbrevation 'HSCANs' denotes 3 hierarchical algorithms. Every point of the chart corresponds to the average value of 100 random graphs.

Modularity for of SCAN produced partitions is usually higher than one of Fast Modularity, but the difference is noticeable only when the number of correctly classified nodes is less than 50%.

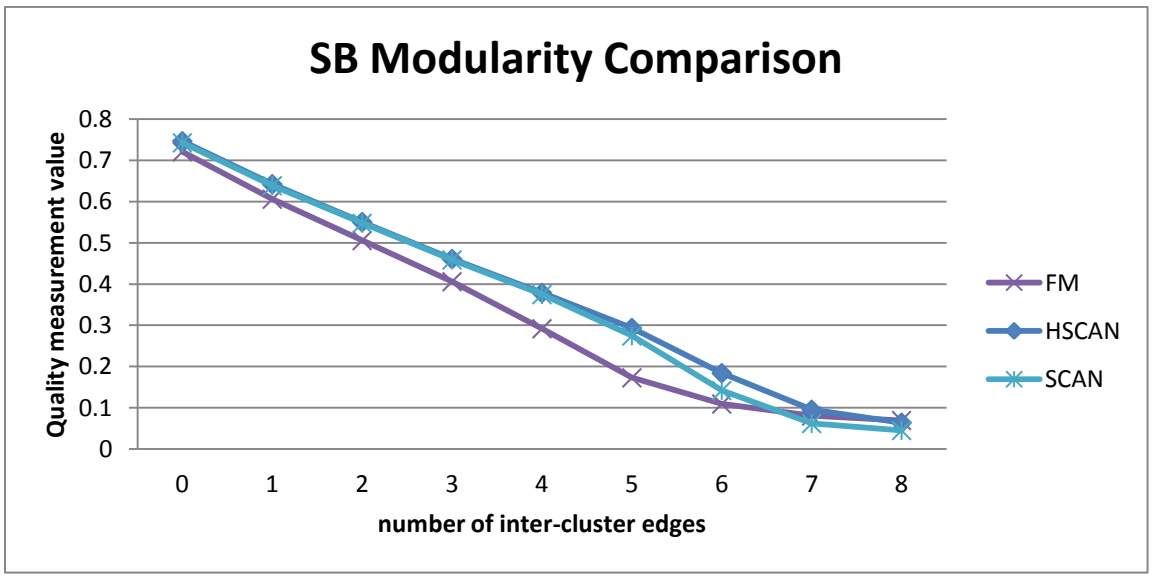

Figure 17 - Similarity-based modularity values for different partitions. Every point of the chart corresponds to the average value of 100 random graphs.

Figure 17 provides similar chart for similarity-based modularity. Values of similarity-based modularity correlate with the number of correctly classified nodes: partitions produced by SCAN-based methods are of the higher quality with respect to this quality function.

Figure 18 - Performance values for different partitions. Every point of the chart corresponds to the average value of 100 random graphs.

Figure 18 demonstrates performance values for the partitions produced by different algorithms. Values of this quality measurement show different relations. We will try to explain them later in this section.

At the next step we decrease weights of the edges, connecting outliers. We change the weights slightly, expecting them to single out special nodes, making them more independent and easy to detect for algorithms.

Weights of intra-cluster edges	0.5
Weights of inter-cluster edges	0.5
Weights of the hub-incident edges	0.2
Weights of the outlier-incident edges	0.2

Table 5 - Graph parameters. Experiment 2.

Overall picture of correctly identified nodes is undistinguishable from figure 13 and we do not provide it again. However the situation with special nodes is indeed better:

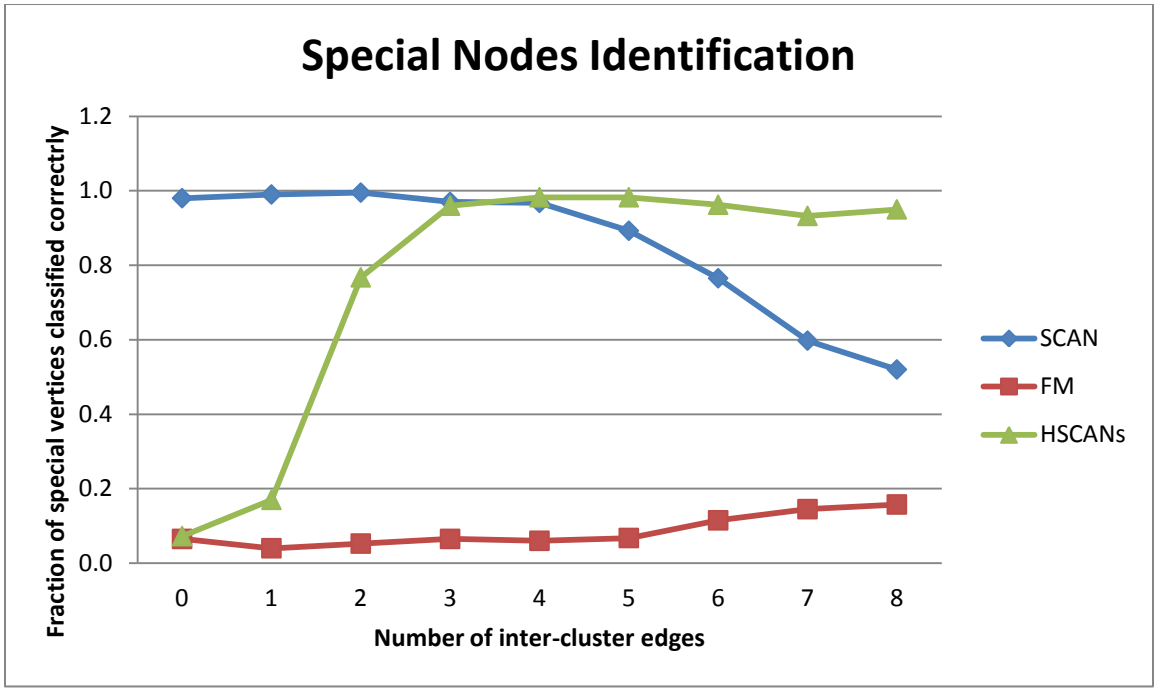

Figure 19 - Fraction of special nodes classified correctly. Every point of the chart corresponds to the average value of 100 random graphs. Abbrevation 'HSCANs' denotes 3 hierarchical algorithms. Their results are identical, so we unite them on the chart.

As we will see from the following experiments FM hardly ever identifies more than 10% of special nodes. The top values are achieved when z_{out} values are high and overall ratio of correctly classified vertices is low, so this increase in 'accuracy' is not likely to be of any value. Comparing Figure 19 with Figure 17 we can notice that SCAN-based algorithms are much better at identifying special nodes, when they are singled out

with weights. SCANs diagram is a little more flat, while HSCANs lift is much more abrupt. These results show that weighted SCAN-based algorithms indeed are sensitive to edge weights and possess certain abilities to identify special nodes.

At the next step we make intra-cluster weights a little bigger in order to emphasize the community structure.

Weights of intra-cluster edges	0.6
Weights of inter-cluster edges	0.5
Weights of the hub-incident edges	0.2
Weights of the outlier-incident edges	0.2

Table 6 - Graph parameters. Experiment 3.

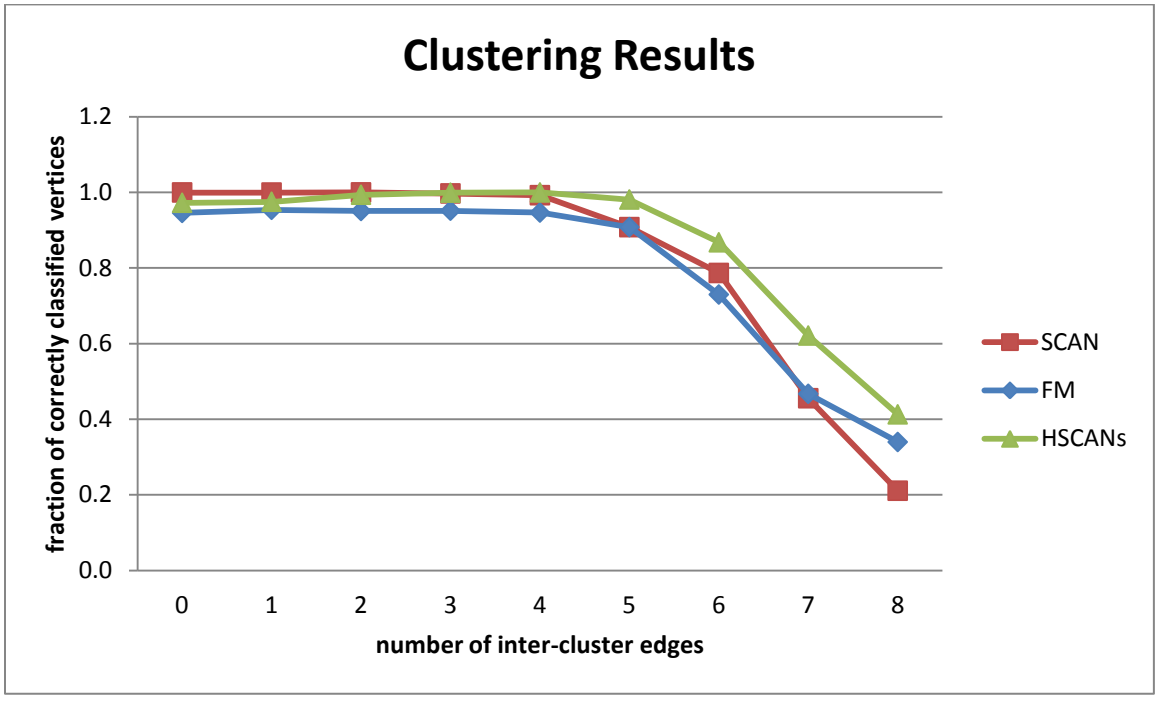

Figure 20 - Fraction of vertices classified correctly after small increase of intra-cluster weights. Every point of the chart corresponds to the average value of 100 random graphs.

Comparison with the figure 13, corresponding to flat weight distribution, reveals that all algorithms improved their performance: we can notice that the slopes of the charts became more abrupt. Thus the algorithms are indeed sensitive to edge weights that now emphasize the community structure of the graph. Moreover Fast Modularity improvement with the change of weights is more significant than one of the other algorithms. This implies that Fast Modularity is more sensitive to edge weights (slight relative change of weights results in noticeable improvement of clustering), which seems quite logical to us, because SCAN-based algorithms also rely a lot on neighbour structure in deriving the vertex similarity value.

A diagram describing special nodes classification as well as quality measurements diagrams are very similar to the one from previous experiment.

We keep increasing the value of intra-cluster weights to strengthen the community structure:

Weights of intra-cluster edges	0.8
Weights of inter-cluster edges	0.5
Weights of the hub-incident edges	0.2
Weights of the outlier-incident edges	0.2

Table 7 - Weight distribution. Experiment 4: increased difference between intra- and inter-cluster edges.

Figure 21 - Fraction of vertices classified correctly. Increased weight difference. Every point on the chart corresponds to the average value of 100 random graphs.

As the weights of inter-cluster edges increase, the community structure is more recognizable for the high values of z_{out}. Thus the chart ends on the right on figure 21 are more flat. Figures show that Fast Modularity algorithm is more sensitive to the edge weights, but neighbour structure orientation makes SCAN-based algorithms better when difference in edge weights is less significant. HSCAN algorithms consequently perform better for the high values of z_{out}, identifying more vertices correctly, while SCAN is a little better when z_{out} is small. Hierarchical algorithms are also bad at classifying special nodes for small z_{out} values.

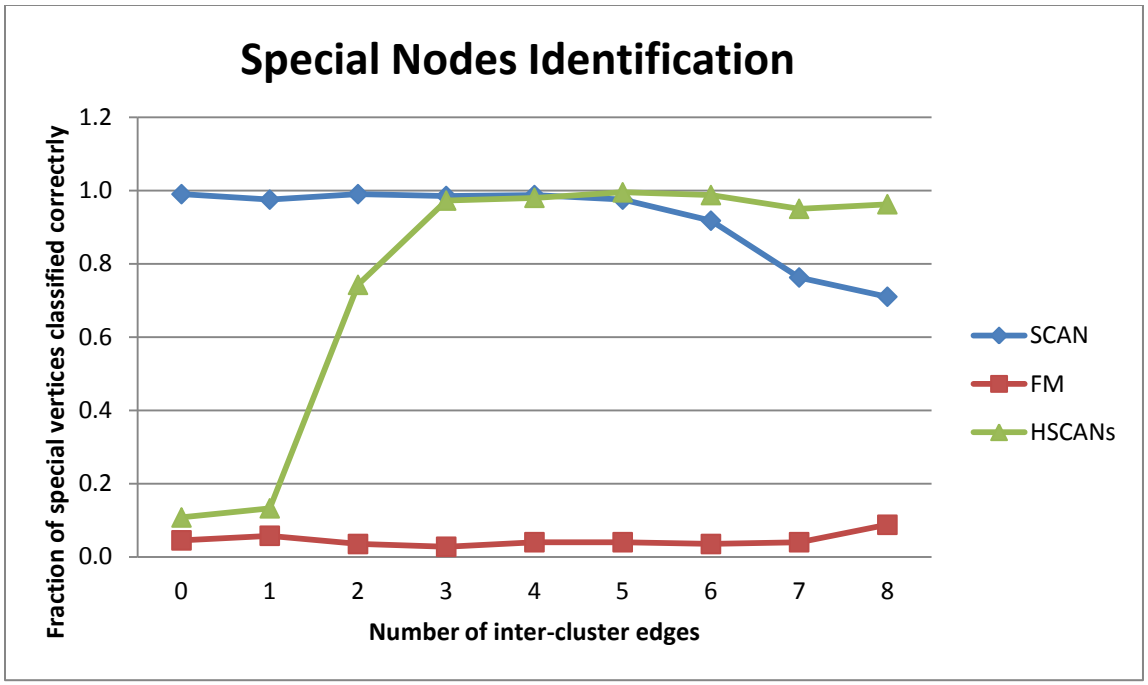

Figure 22 - Fraction of special nodes identified correctly. Every point on the chart corresponds to the average value of 100 random graphs.

It is hard to tell how relevant the right side of these charts is, and how valuable is high degree of special nodes classified correctly while the overall ratio of correct vertices is low (keeping in mind that some algorithms tend to produce singleton clusters for high z_{out} values, which does not benefit the overall classification, but indeed identifies most of special nodes). This issue will require some attention in the future.

At the next stages we keep strengthening intra-cluster edges, while this time we also decrease inter-cluster edges and 'dim' special nodes.

Weights of intra-cluster edges	0.9
Weights of inter-cluster edges	0.4
Weights of the hub-incident edges	0.4
Weights of the outlier-incident edges	0.4

Table 8 - Weight distribution. Experiment 5: increased difference between intra- and inter-cluster edges.

Figure 23 - Strengthening of community structrue yields better partitions. Every point on the chart corresponds to the average value of 100 random graphs.

We see that results are consistent: SCAN is better for small z_{out}, hierarchical SCAN are better for big z_{out}. Fast Modularity is a little behind both of them everywhere, except for the range of high z_{out}, when it is more accurate than SCAN.

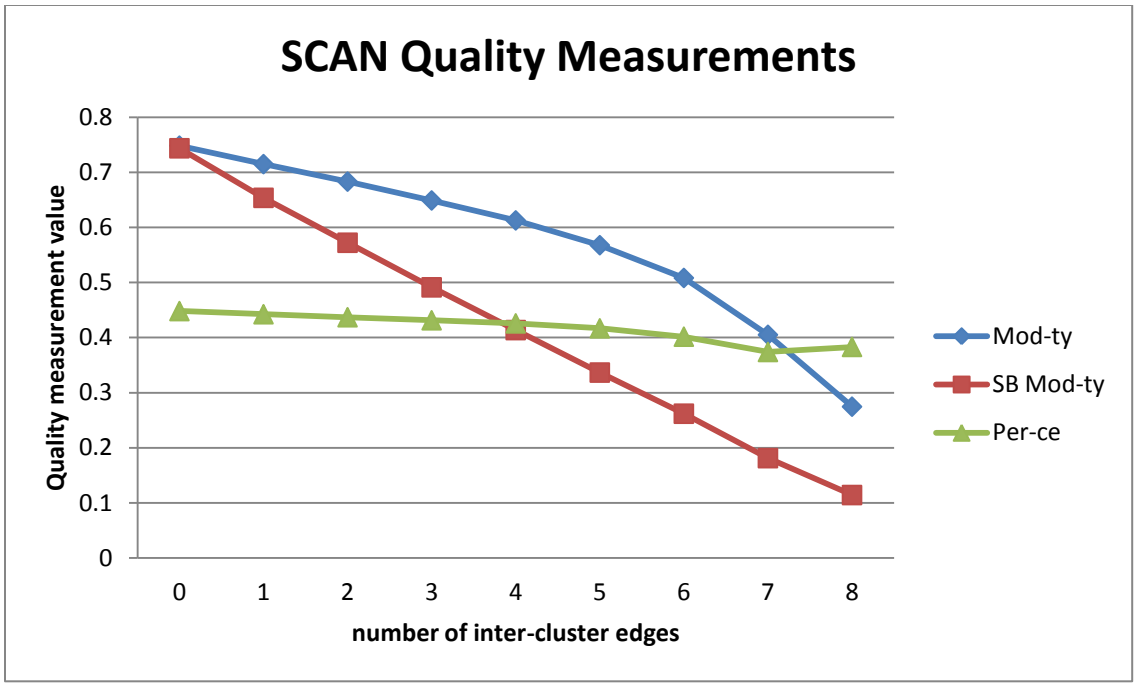

Figure 24 - Graph clustering quality measurements. Every point on the chart corresponds to the average value of 100 random graphs.

Charts on figure 21 are representative for partitions produced by all studied algorithms. With the strengthening of community structure Newman's modularity tends to become more flat, while similarity based modularity turns into almost linear dependency. We will provide some analysis of these measurement further.

For the final experiment we increase the intra/inter -weights ratio to the maximum.

Weights of intra-cluster edges	1.0
Weights of inter-cluster edges	0.3
Weights of the hub-incident edges	0.3
Weights of the outlier-incident edges	0.3

Table 9 - Final experiment. Weights distribution. Maximum intra/inter ratio.

Figure 25 - The ratio of correctly classified vertices does not go lower 85%. Every point on the chart corresponds to the average value of 100 random graphs.

At the bigger scale we finally can notice some insignificant changes for AHSCAN with Newman's modularity. It is recognizable only in 'extreme' conditions of the community structure being express only in edge weights. Special nodes results remain the same.

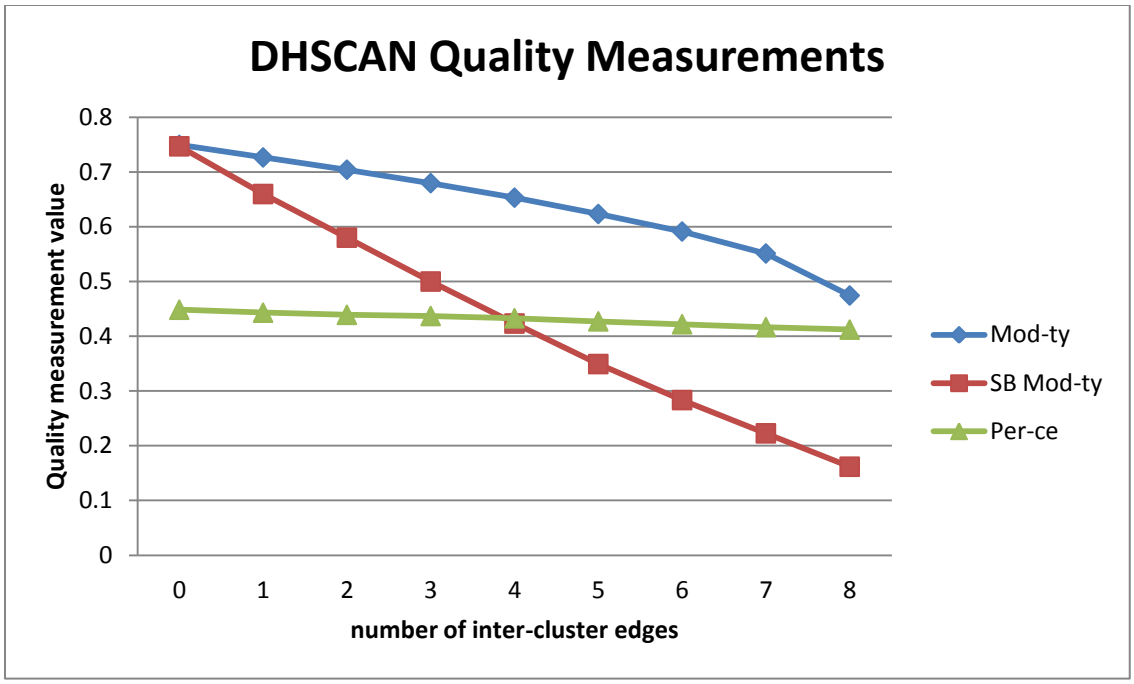

Figure 26 - Quality measurements of DHSCAN produced partitions. Every point on the chart corresponds to the average value of 100 random graphs.

Figures 15 and 26 show correlations between clustering quality measurements for somewhat extreme cases. Quality measurements charts are not very descriptive. But together with comparison charts we are able to make some conclusions, that are provided further.

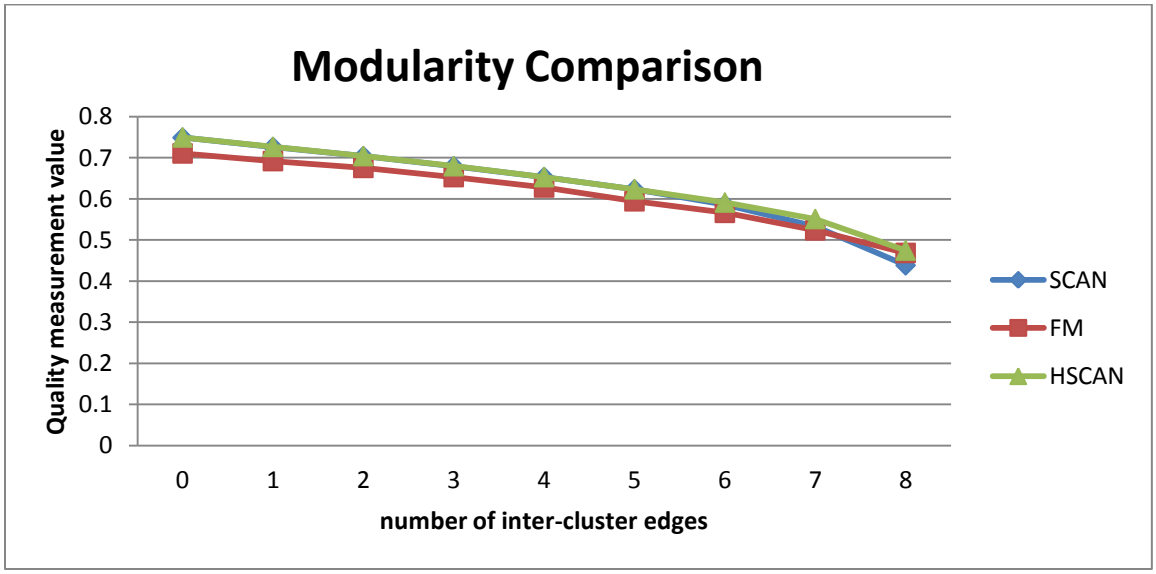

Figure 27 - Modularity values for different partitions

Figure 27 demonstrates modularity values for the partitions of graphs with this type of weight distribution. We did not provide such charts for previous experiments, except for the first one because of their very high similarity. Overall dynamic is such that SCAN-based algorithms produce better partitions in terms of modularity.

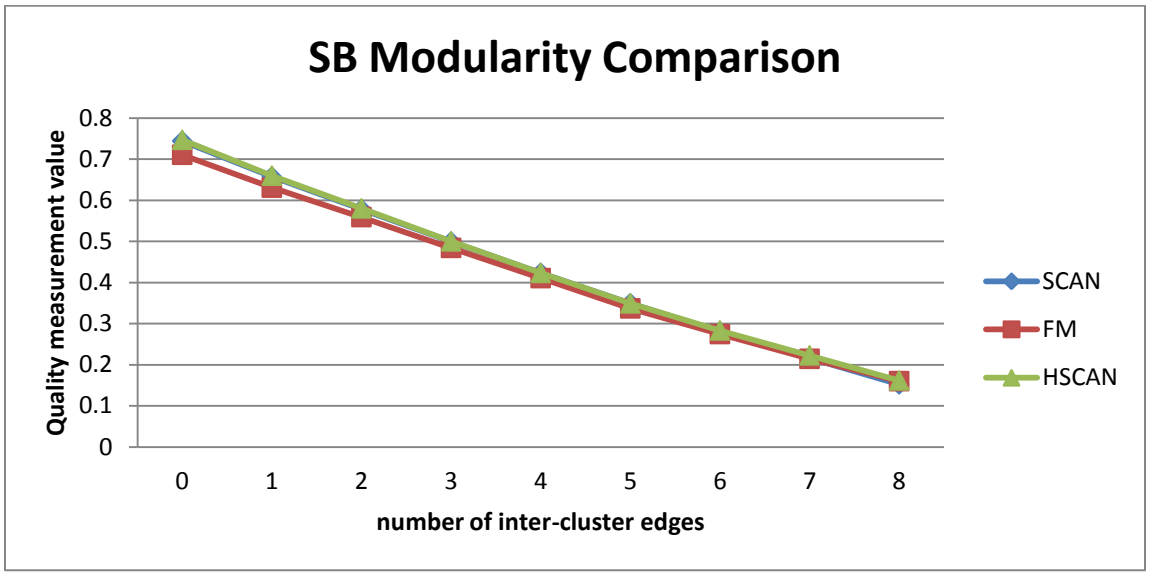

Figure 28 - Similarity-based modularity for different partitions.

Figure 28 demonstrates charts for similarity-based modularity. SCAN-based algorithms results are a little better than ones of Fast Modularity.

On the charts we can see that modularity and similarity-based modularity correlate most of the times and they never contradict when measuring meaningful graph partitions. We noticed them contradicting only for the case of graphs with flat weight distribution for the maximum values of z_{out} when the fraction of correctly identified nodes is far less than 50%.

Figure 29 - Performance values for different partitions.

Figure 29 shows performance values of different partitions. For this experiment performance values correlate with the number of correctly classified nodes, but in general performance values are rather contradictory. Quality functions charts allow us to conclude that performance quality measurement is not suitable for our purposes, at least with the type of graphs used in this experimental setup. For this type of graphs it produces barely meaningful and hardly predictable values. Its value goes up as special nodes or the nodes mistakenly identified as such are placed in singleton clusters - the number of nonexistent inter-cluster weights is almost equal to the number of nodes in the graph. Thus performance happens to be a bad choice for a reference quality measurement function.

Weighted SCAN-based algorithms perform very well on this type of graphs and similarity-based modularity shows itself as a valid quality measure. Our experiments showed that new weighted algorithms are indeed sensitive to edge weights, weights indeed contribute to better clustering.

IV.1.2. Power laws and preferential attachment

We strived to make our random graphs as realistic as possible and studied certain aspects inherent to real graphs, including the number of edges to number of vertices ratio. We examined a number of real world graphs, mentioned in various clustering papers, including the most famous works by Zachary [38], Newman and Clauset [27], [25], [5] and many others [34]. In most cases the edges to nodes ratio falls in a range from 2 to 7 with some curious exceptions: for instance a specific graph used by Dorogovtsev in [9] describes a word web and has average degree of 72. Targeting the most common graph properties, we generated graphs with edges to nodes ratio varying from 2 to 7. We used a generator of random graphs based on Eppstein and Wang's approach described in [11], assigning weights to edges based on clustering coefficient of the nodes they connect. The charts below show the average values of quality measurements of the partitions produced by different clustering algorithms. We also used a random graph generator to provide the 'ground level' for the quality measurements comparison. Numbers on horizontal lines of the charts below correspond to hundreds of edges in graphs. Table 10 provides the average standard deviation for a typical experiment on this graph type.

Number of vertices = 50

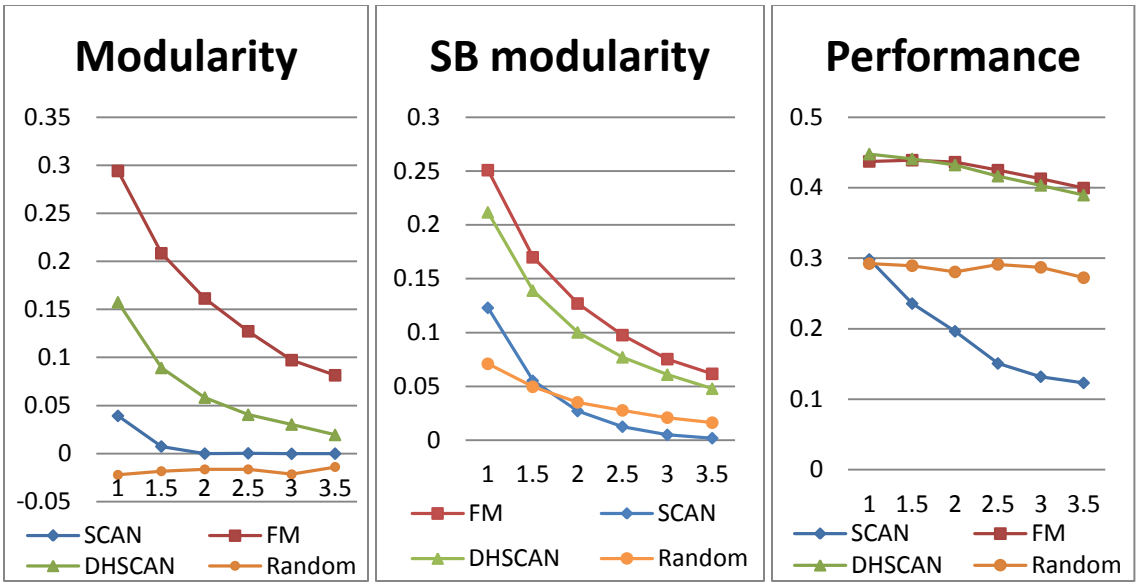

Figure 30 - Quality measurements for Eppstein power-law random graphs with 50 vertices. Horizontal line marks hundreds of edges. Every point on the chart corresponds to the average value of 100 random graphs.

	FM	SCAN	DHSCAN	AHSCANs	AHSCANn	Random
Modularity	0.015	0.026	0.03	0.03	0.03	0.017
SB Modularity	0.01	0.012	0.013	0.013	0.013	0.015
Performance	0.007	0.015	0.016	0.016	0.016	0.08

Table 10 - Standard deviation values

Modularity and performance for random partitions are represented with almost straight lines, which can be considered as an indirect proof of quality measurements validity, although performance values are still confusing. Modularity, originally claimed by authors as falling in range [0,1] [27] takes negative values, although close to zero. Similarity-based modularity is not so stable, which may imply that it is not so reliable quality measurement.

Extremely poor results for SCAN can be explained by a lack of 'calibration': pretest runs in order to identify the best values of the input parameter. As the number of edges increases the algorithm might require the change of input parameter value. This

requirement of setting the proper parameter value, which is different for different types of graphs is the serious algorithms drawback.

Only similarity-based modularity allows to identify minor differences in behaviour of agglomerative hierarchical SCAN with Newman's modularity as a quality measure. All other parameters are almost identical to DHSCAN.

Number of vertices = 100

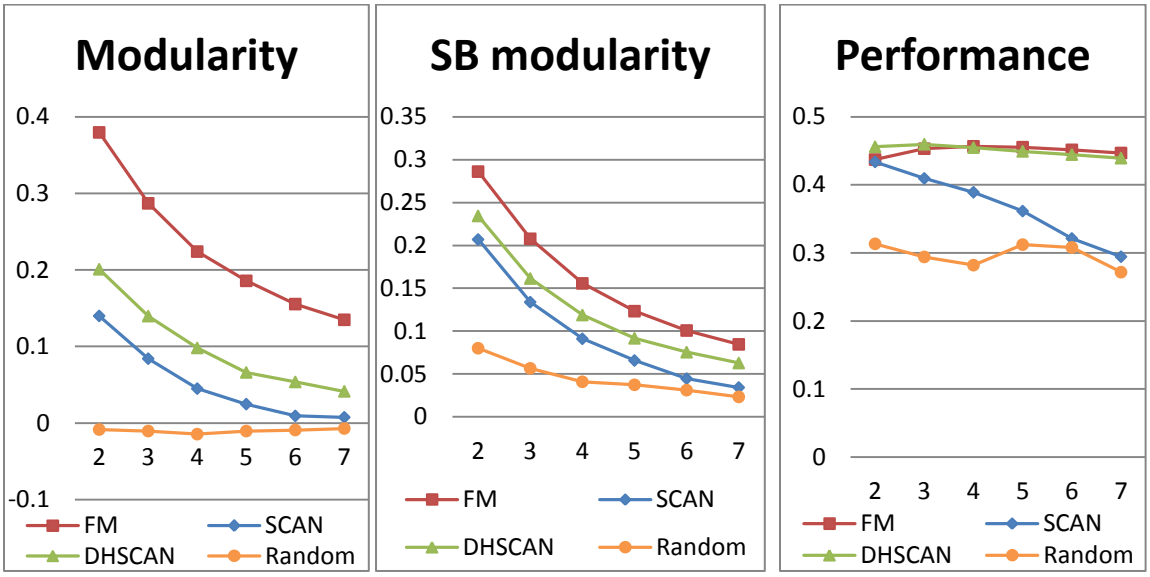

Figure 31 - Quality measurements for Eppstein power-law random graphs with 100 vertices. Horizontal line marks hundreds of edges. Every point on the chart corresponds to the average value of 100 random graphs.

As we increase the number of vertices the overall picture stays almost the same. As the number of edges increases all the methods tend to fail. Fast Modularity algorithm turns out to be more suited for this type of graphs. We also notice a strange increase in performance value of SCAN results, but for this set of graphs starting values of modularity also increased more than twice for partitions produced by SCAN. Not sufficient amount of experiment can be a possible explanation for such deviations.

Number of vertices = 200

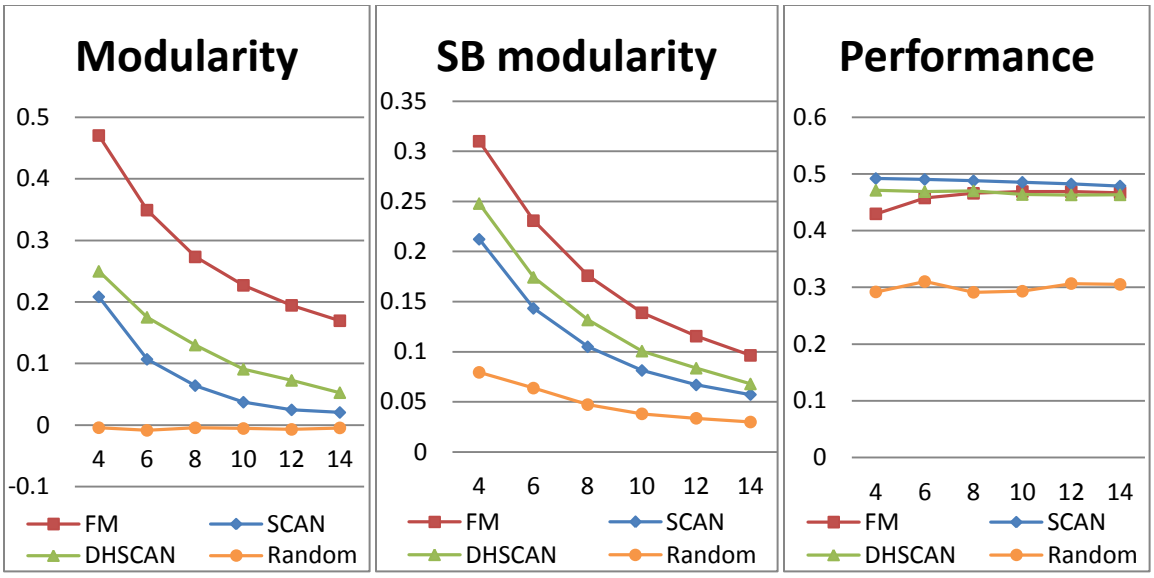

Figure 32 - Quality measurements for Eppstein power-law random graphs with 200 vertices. Horizontal line marks hundreds of edges. Every point on the chart corresponds to the average value of 100 random graphs.

Overall dynamics stays the same, though starting values of all quality measurements become bigger. It turns out that the larger the graph, the more pronounced its community structure is for the algorithms. Overall poor results of SCAN-based methods can be explained by 'non-social' structure of this types of graphs: achieved by Fast Modularity clusters, presented on the Figure 32, are rather tightly connected to each other and in terms of structural similarity many of them are to be clustered together. Figure 32 presents typical Eppstein Power Law graph with 200 nodes and 400 edges clustered by Fast Modularity algorithm.

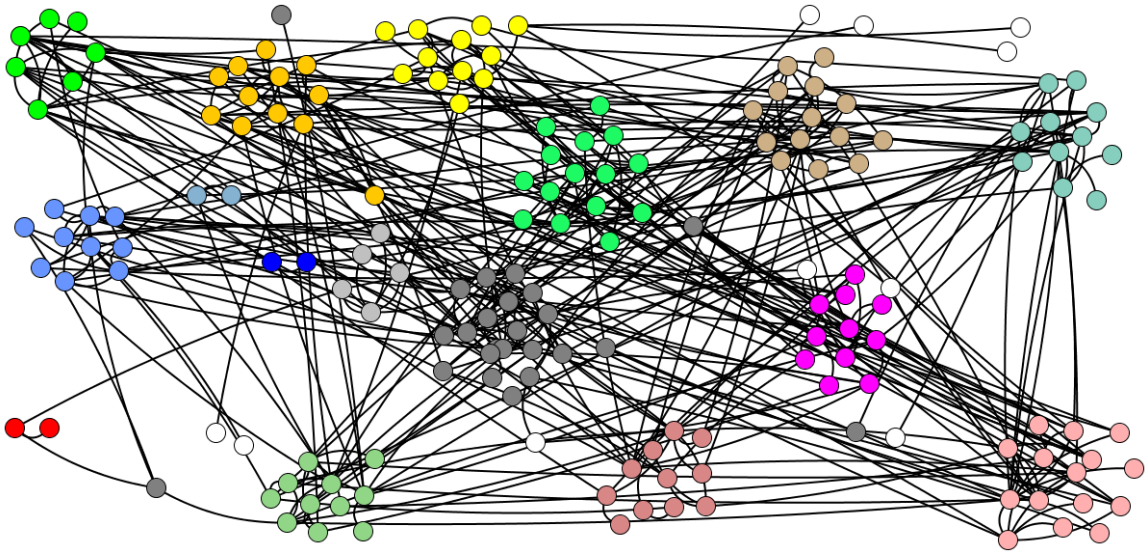

Figure 33 - A typical Eppstein Power Law random graph with 200 nodes and 400 edges. Fast Modularity clustering algorithm identified 13 relatively large clusters.

Figure 33 shows the same graph clustered by DHSCAN, for which high degree of inter-cluster connection turns out to be challenge. DHSCAN identifies only two big clusters in the same graph, while Fast Modularity algorithm finds 13 smaller ones.

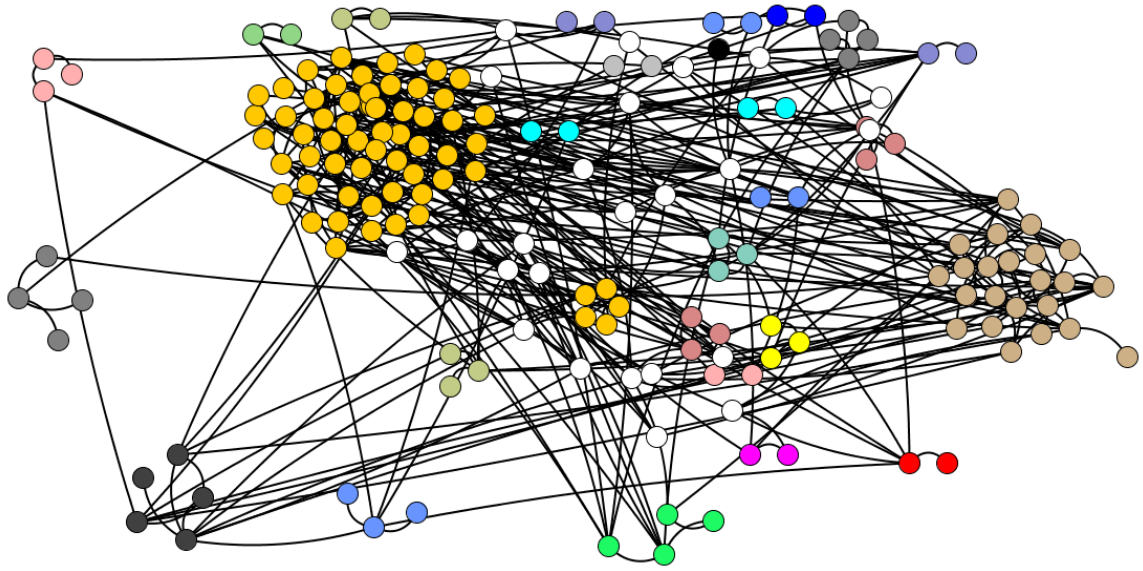

Figure 34 - Same graph clustered by DHSCAN. Social structure is hard to separate.

Results for Preferential attachment graph generator are surprisingly similar, so we do not provide them.

This set of experiments demonstrates that SCAN-based algorithms are not suited for any kind of graphs, which was implied to by their definition. Neighbourhood structure inherent to social networks is a prerequisite for a successful SCAN-based algorithms execution.

IV.1.3. Real dataset (ENRON)

Enron data set consists of e-mail database of Enron company, made public by the Federal Energy Regulatory Commission after company's bankruptcy [30]. For our experiments, similarity to Falkowski in [13], we counted the number of reciprocal interactions between Enron company workers, thus separating broadcasting messages that do not contribute to the real network structure. We took a time frame of few months to gather sent and received messages and build a graph. As a result we got 442 vertices connected by 710 edges.

	FM	SCAN	DHSCAN	AHSCANs	AHSCANn
Modularity	0.76	0.60	0.25	0.25	0.63
SB modularity	0.68	0.25	0.47	0.47	0.32
Performance	0.45	0.22	0.28	0.28	0.47

Table 11 - Quality functions values for different partitions of ENRON data set

Fast Modularity produces the best result with respect to quality measurements, the graph partition is shown on the figure 35.

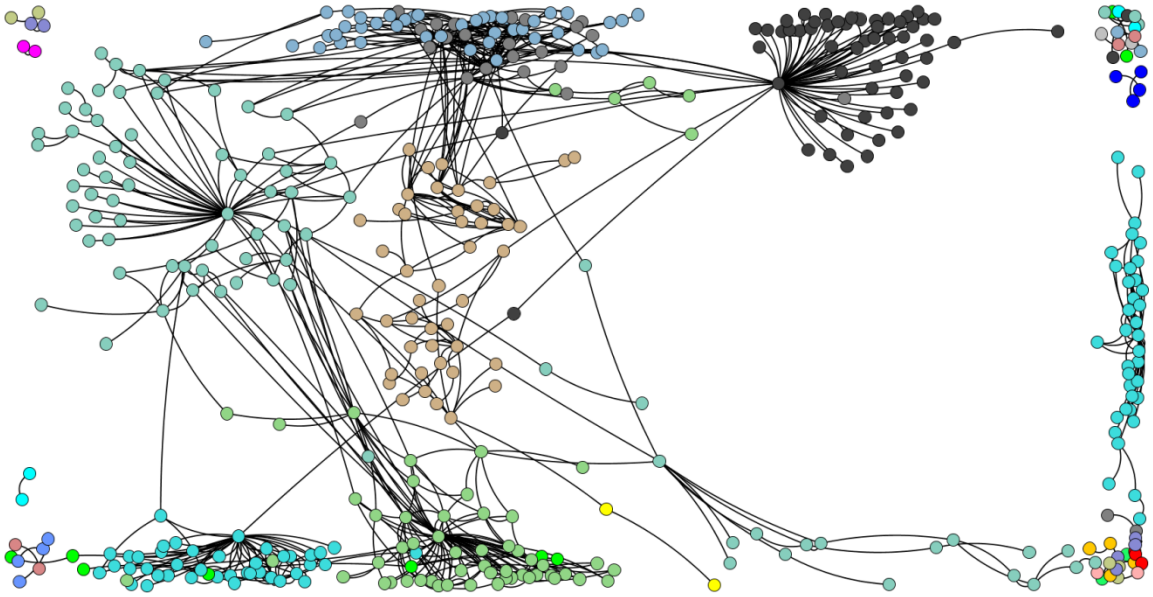

Figure 35 - Fast Modularity clustering of Enron dataset

The next best result is produced by AHSCAN with Newman's modularity as quality measurement. The result is shown on figure 36.

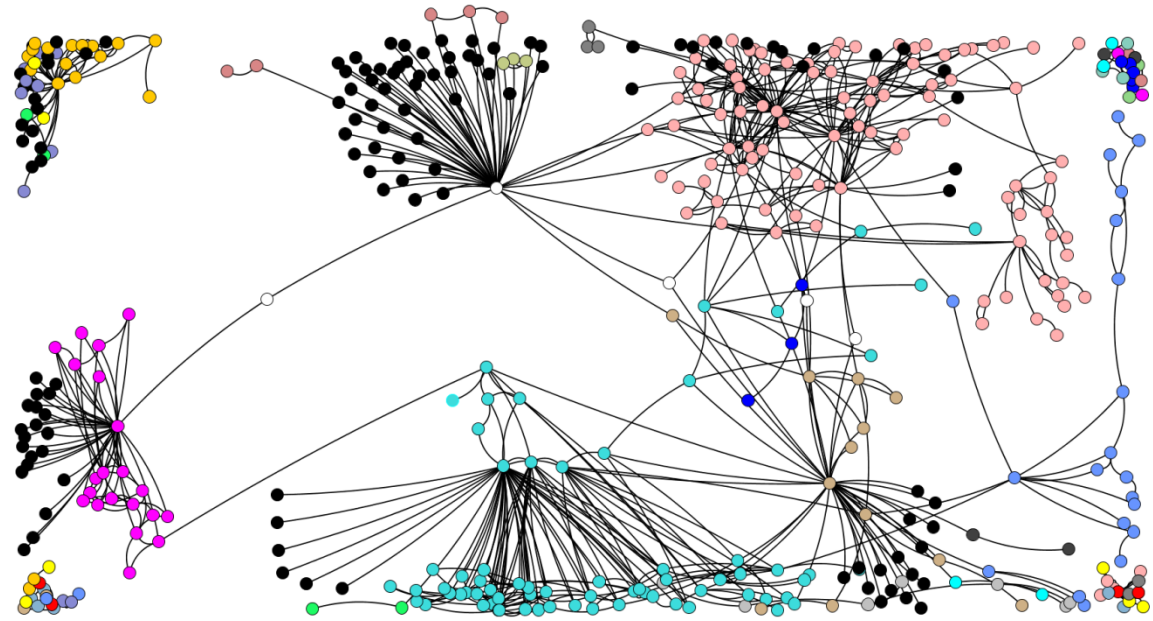

Figure 36 - AHSCAN clustering of Enron dataset

As we can see, comparing two partitions, the result produced by AHSCAN can turn our very much more valuable for the sake of social network analysis as it produces hubs and outliers (black and white nodes on the picture).

SCAN results are rather high with respect to quality measurements, but identifies less clusters than AHSCAN. Figure 37 shows the produced partition. Where special nodes identification can be seen in action: a bunch of black nodes identify outliers, obviously individuals that were not active with their e-mails.

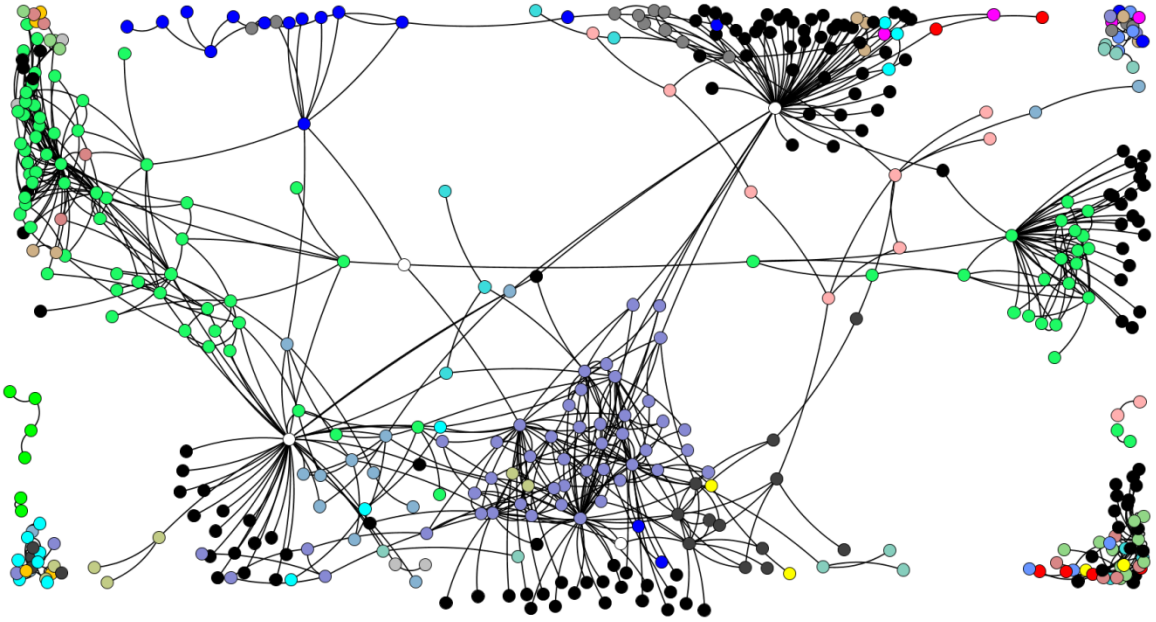

Figure 37 - SCAN clustering of Enron dataset

CHAPTER V

CONCLUSIONS AND FUTURE WORK

We have shown that algorithms complexity is not affected by our modifications. The experiments proved extended SCAN-based algorithms to be able to identify community structure in weighted graphs, to be competing with the top-rated Fast Modularity method and producing better results (identifying more nodes correctly) for at least one type of graphs: planted l-partition model, in other words, graphs with pronounced community structure, which describes favourable conditions for SCAN-based algorithms applications. Extended SCAN-based algorithm do produce better partitions in terms of 2 of 3 quality measurement functions for that type of graphs. The third quality function (performance) proved to be unfit for our purposes and produced contradictory results. Results for other types of graphs with unknown community structure are not as promising, which makes it worth trying proposed algorithms on more graphs in future experiments. New algorithms were validated on generated random graphs with predictable clustering structure. Getting some real datasets with known existing partition and checking our methods accuracy on them is important part on the future work as well as possible usage of word graphs, described by Dorogovtsev and Mendes in [9], for that purposes.

The experiments with hierarchical SCAN algorithms (DHSCAN, AHSCAN) proved that similarity-based modularity does possess an ability to detect special nodes, but it appears somewhat limited and unpredictable. Researching, tuning and improving of similarity based modularity or any other existing clustering quality measurement, or

development of a new one that will take singleton clusters and special nodes into account is interesting research topic.

With the help of Eppstein power law random graph generating framework we were able to find out about serious limitations of SCAN-based methods, although no advantages were identified. Applying aspects of social networks, some of which are pointed and described by Jin et al. in [22], to random graphs can possibly make random graphs more suitable for 'social-oriented' SCAN-based algorithms.

Fast Modularity being considered to be one of the top-rated graph clustering algorithms, is an optimization of Newman's algorithm [26], adapted for very large networks. The authors mention that 'hierarchies generated by these two versions' of the Modularity algorithm will be slightly different 'as a result of the differences in how ties are broken for the maximum element in a row' [5]. This might imply that the quality of the solutions produces by a fast version of the algorithm lack some accuracy. Thus as a future work we would consider comparing weighted versions of SCAN-family algorithms with original 'slow' Newman's algorithm.

We showed that modifications proposed in this work do not influence the complexity of the algorithms, and we did not carry any measurements related to the time of execution, etc. Moreover the quality check of algorithms running time requires elaborate implementation of all of them. It is known that algorithm theoretical complexity might be very different from implementation complexity. It may also depend on the programming language used, level of caching implemented, etc. This is a good topic for a vast future research.

Original structural similarity used by the authors of SCAN can have values in range [0, 1]. It is very convenient and intuitive. Our new measure, structural similarity extended with edge weight can take values in range [0, 2] which is less intuitive. The idea of its normalization can be considered for the future work as well. Moreover extended structural similarity does not include both components (neighbourhood similarity and edge weight) in equal proportion, it gives more 'weight' to neighbourhood structure. One more possible weak side of newly proposed extended structural similarity is the necessity of weights normalization: if a network has very few 'heavy' edges, overall role of other edges will be evened by the normalization. Thus tuning, finding balance or a way to adjust the influence of different components of similarity seems to be promising idea for future exploration.

The idea of special nodes that do not belong to any cluster, and especially hubs, that connect different clusters and thus, in other words, may belong to few clusters can be a key to the problem of overlapping communities and fuzzy clusters, some information on which can be found in survey of Santo and Fortunato [28].

REFERENCES

[1] Albert Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[2] Ulrik Brandes and Thomas Erlebach. Network analysis: Methodological foundations. In Ulrik Brandes and Thomas Erlebach, editors, *Network Analysis*, volume 3418 of *Lecture Notes in Computer Science*, pages 1–6. Springer Berlin / Heidelberg, 2005. 10.1007/978-3-540-31955-9_1.

[3] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38, June 2006.

[4] A. Chertov, Z. Kobti, and S.D. Goodwin. Weighted scan for modeling cooperative group role dynamics. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 17 –22, aug. 2010.

[5] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(6):066111, December 2004.

[6] Anne Condon and Richard M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18:116–140, March 2001.

[7] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, 2007.

[8] C.H.Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and H.D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 107 –114, 2001.

[9] S. N. Dorogovtsev and J. F. F. Mendes. Language as an evolving word web. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1485):2603–2606, 2001.

[10] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27 – 64, 2007.

[11] David Eppstein and Joseph Wang. A steady state model for graph power laws. *CoRR*, cs.DM/0204001, 2002.

[12] B. S. Everitt. Cluster analysis: A brief discussion of some of the problems. *The British Journal of Psychiatry*, 120(555):143–145, 1972.

[13] Tanja Falkowski. *Community Analysis in Dynamic Social Networks*. PhD thesis, Otto-von-Guericke-University, Magdeburg, 2009.

[14] Zhidan Feng, Xiaowei Xu, Nurcan Yuruk, and Thomas Schweiger. A novel similarity-based modularity function for graph partitioning. In Il Song, Johann Eder, and Tho Nguyen, editors, *Data Warehousing and Knowledge Discovery*, volume 4654 of *Lecture Notes in Computer Science*, pages 385–396. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-74553-2_36.

[15] Gary William Flake, Robert E. Tarjan, and Kostas Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004.

[16] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. In Ira Gessel and Gian-Carlo Rota, editors, *Classic Papers in Combinatorics*, Modern Birkhдuser Classics, pages 243–248. Birkhдuser Boston, 1987. 10.1007/978-0-8176-4842-8_15.

[17] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[18] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):pp. 551–570, 1961.

[19] Roger Guimera and L. A. N. Luis. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, February 2005.

[20] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985. 10.1007/BF01908075.

[21] Kenneth E. Iverson. A programming language. In *Proceedings of the May 1-3, 1962, spring joint computer conference*, AIEE-IRE '62 (Spring), pages 345–351, New York, NY, USA, 1962. ACM.

[22] Emily M. Jin, Michelle Girvan, and M. E. J. Newman. Structure of growing social networks. *Phys. Rev. E*, 64:046132, Sep 2001.

[23] Donald E. Knuth. Two notes on notation. *Am. Math. Monthly*, 99:403–422, May 1992.

[24] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):pp. 167–256, 2003.

[25] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, Nov 2004.

[26] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, Jun 2004.

[27] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, Feb 2004.

[28] Santo and Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010.

[29] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173 – 182, 2004. <ce:title>Discrete Mathematics and Data Mining</ce:title>.

[30] Jitesh Shetty and Jafar Adibi. The enron email dataset database schema and brief statistical report. Technical report, 2004.

[31] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888 –905, aug 2000.

[32] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):pp. 425–443, 1969.

[33] D J Watts and S H Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):440–442, 1998.

[34] F. Wu and B.A. Huberman. Finding communities in linear time: a physics approach. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38:331–338, 2004. 10.1140/epjb/e2004-00125-x.

[35] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A. J. Schweiger. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 824–833, New York, NY, USA. ACM, 2007.

[36] Nurcan Yuruk, Mutlu Mete, Xiaowei Xu, and Thomas A. J. Schweiger. A divisive hierarchical structural clustering algorithm for networks. *Data Mining Workshops, International Conference on*, 0:441–448, 2007.

[37] Nurcan Yuruk, Mutlu Mete, Xiaowei Xu, and Thomas A.J. Schweiger. Ahscan: Agglomerative hierarchical structural clustering algorithm for networks. *Social Network Analysis and Mining, International Conference on Advances in*, 0:72–77, 2009.

[38] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):pp. 452–473, 1977.

VITA AUCTORIS

Anton Chertov was born in 1985 in Simferopol, Crimea, Ukraine. He graduated from high school in 2002. From there he went on to Moscow Aviation Institute (State Technology University) where he obtained a degree in Software Engineering in 2008. He is currently a candidate for the Master's degree in Computer Science at the University of Windsor and plans to graduate in Winter 2012.